TARTU ÜLIKOOL

Majandusteaduskond

Karl-Kristjan Luberg

TÖÖ- JA PALGAARVESTUSTARKVARADE ARENDAMISE PRAKTIKA

Bakalaureusetöö

Juhendaja: majandusarvestuse dotsent Kertu Lääts

Tartu 2016

Soovitan suunata kaitsmisele ….....

(juhendaja allkiri)

Kaitsmisele lubatud …..... 2016. a.

…….…….…….. õppetooli juhataja ……..……

(õppetooli juhataja nimi ja allkiri)

Olen koostanud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, põhimõttelised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

……

(töö autori allkiri)

SISUKORD

SISSEJUHATUS

Tingituna ettevõtete globaliseerumisest, infohulkade ning konkurentsi pidevast suurenemisest on ettevõtetel vaja aina rohkem kulusid optimeerida. Optimeerimine eeldab olemasolevate protsesside ja protseduuride läbimist efektiivsemalt. Paratamatult tuleb igal ettevõttel puutuda kokku töö- ja palgaarvestusega ning sellest tulenevalt on ettevõte huvitatud arvestusega seotud protsesside efektiivsemaks muutmisest tarkvarade abil. Antud probleeme lahendada aitavad tarkvarad vajavad pidevat arendamist. Käesolev bakalaureusetöö püüab piiritleda ja analüüsida olemasolevaid lahendusi ning määratleda enim kasutatud praktika töö- ja palgaarvestustarkvarade edasiarendamiseks.

Autor oli motiveeritud bakalaureusetööd kirjutama, sest tegeles hobikorras töö- ja palgaarvestustarkvara Wemply arendamisega. Senine Wemply tarkvara arendamine ei ole toimunud süsteemselt ja mõtestatult. Käesoleva bakalaureusetööga soovib autor teiste tarkvaraarendajate kogemustest lähtuvalt arendada edasi Wemply tarkvara.

Bakalaureusetöö eesmärk on määratleda Eesti tarkvaraettevõtete seas enim kasutatud töö- ja palgaarvestustarkvara arendamise praktikad. Enim kasutatud praktikale tuginevalt on võimalik kujundada paremad arendamise võimalused Wemply tarkvarale. Samas võivad töö tulemused olla huvipakkuvad ka laiemale tarkvaraarendusega tegelejate ringile.

Eesmärgi saavutamiseks on autor pannud kirja järgmised uurimisülesanded:

1) selgitada arvestuse infosüsteemide automatiseerimise vajalikkust,
2) anda ülevaade töö- ja palgaarvestustarkvara arendamise meetoditest ja tööetappidest,
3) anda ülevaade töö- ja palgaarvestustarkvarade arendamist mõjutavatest teguritest,
4) anda ülevaade töö- ja palgaarvestustarkvaradest Eestis,
5) analüüsida kogutud tarkvaraarenduse andmeid,

6) tuua välja enim kasutatud praktika töö- ja palgaarvestustarkvarade arendamiseks,
7) edasi arendada Wemply't enim kasutatud praktikale tuginedes.

Bakalaureusetöö koosneb kahest peatükist – teooria ja empiiriline osa.

Teooria oluline osa on arvestuse infosüsteemide automatiseerimise vajalikkuse selgitamine. Antakse ülevaade töö- ja palgaarvestuse automatiseerimise teoreetilistest alustest ja sellega seonduvatest mõistetest. Oluline osa on antud teemas tarkvaraarendusel ja seda mõjutavatel teguritel. Antakse ülevaade tarkvaraarenduse meetoditest ja tööetappidest. Tutvustatakse tarkvaraarendust mõjutavaid tegureid. Peamine erialakirjandus, millele tuginetakse, hõlmab arvestuse infosüsteemide automatiseerimise alaseid ning tarkvaratehnika teoreetilisi aluseid käsitlevaid teadusartikleid ja raamatuid.

Empiirilises osas tutvustatakse turul olevaid töö- ja palgaarvestustarkvarasid. Tutvustatud töö- ja palgaarvestustarkvarade esindajatega viiakse läbi intervjuud. Intervjuude käigus kogutud andmete põhjal määratletakse enim kasutatud praktika töö- ja palgaarvestustarkvarade arendamiseks. Määratletud praktikat kasutatakse Wemply edasiarenduseks.

Bakalaureusetöös on kasutatud erinevate autorite teemakohaseid materjale ning veebis vabalt kättesaadavaid e-raamatuid töö- ja palgaarvestuse teemadel. Oluline roll on erinevatel teema konteksti kuuluvate tarkvarade koduleheküljedel. Töö on oluline kõigile, kes tegelevad töö- ja palgaarvestustarkvarade arendamisega või huvituvad töö- ja palgaarvestusetarkvarade arendamise praktikast.

Antud bakalaureusetööd iseloomustavad märksõnad: töö- ja palgaarvestus, tarkvara ja tarkvaraarendus.

Bakalaureusetöö autor tänab intervjueerituid ja intervjuudes osalenud ettevõtete töötajaid antud vastuste eest ning töö juhendajat Kertu Läätse tema professionaalse juhendamise eest.

1. TÖÖ- JA PALGAARVESTUSTARKVARA ARENDAMISE TEOREETILISED ALUSED

1.1. Arvestuse infosüsteemide automatiseerimise vajalikkus

Töö- ja palgaarvestuse automatiseerimine algab mõistest arvestuse infosüsteem ning veelgi laiemalt infosüsteem. Lähenedes induktiivselt antud teemale luuakse selge ülevaade teemaga seonduvatest mõistetest ning definitsioonidest. Käesolevas alapeatükis seletab autor lahti arvestuse infosüsteemide mõiste ja nende automatiseerimise vajalikkuse. Automatiseerimine hõlmab vastava tarkvara arendamist protsessi automatiseerimiseks ja sellest tulenevalt tuuakse sisse tarkvaraarenduse mõiste.

Infosüsteem on raamistik, mille kaudu erinevad ressursid on koordineeritud sisenditest ehk andmetest looma väljundeid ehk informatsiooni saavutamaks ettevõtte eesmärke. Andmete töötlemise tulemus on informatsioon. (Wilkinson 1991: 4)

Arvestuse infosüsteem on infosüsteem, kus on integreeritud finants- ja juhtimisarvestus. Tegu on ettevõtte kõige suurema alamsüsteemiga. Finantsarvestuse peamine eesmärk on andmete kogumine ettevõtte majandustehingute kohta ettevõtte väliste infotarbijate huvidele vastavalt. Seadustega ettenähtud aruandeid vajavad väljaspool ettevõtet asuvad infokasutajad, kuid neid aruandeid kasutatakse ka ettevõtte sees. (Pärl 2006: 2) Juhtimisarvestuse eesmärk on anda juhtkonnale vajalikku informatsiooni paremaks juhtimisotsuste tegemiseks ehk sihiks on sisemised infotarbijad.

Arvestuse infosüsteemid koguvad, salvestavad, hoiustavad ja töötlevad andmeid saamaks väljundina informatsiooni otsuste langetamiseks (Accounting … 2016). Töö- ja palgaarvestus hõlmab endas palju erinevaid ülesandeid nagu näiteks arvete ja palkade üle järjepidamine. Taolised rutiinsed tehingute läbitöötamised on ettevõtte jaoks suur kulu ja sellise kulu vähendamine oli peamiseks motivatsiooniks arendamaks arvestuse

infosüsteeme. Arvestuse infosüsteemide arendamise eesmärk oli automatiseerida sellised tegevused. Kuigi andmehulgad on suhtelised, siis algoritmid nende andmehulkade töötlemiseks on suhteliselt lihtsad ja sarnanevad erinevate ettevõtete lõikes. Suurel hulgal arvandmete töötlemine läbi lihtsate algoritmide oli ideaalne rakendus algelistele andmetöötlustarkvaradele ja sellest tulenevalt majandusarvestuse automatiseerimine oli paljude ettevõtete jaoks esimene kokkupuude arvutitega. (Wilson 1992: 65)

Mõned arvestuse infosüsteemide ülesanded on rohkem automatiseerimise poolt mõjutatud kui teised. Lihtsate algoritmidega arvandmete töötlemiseks, nagu näiteks raamatupidamine/palgaarvestus, on arendatud valmis tarkvarad. See tähendab, et eelnevalt manuaalselt täidetud ülesanded on nüüd tarkvara poolt automaatselt täidetavad. Teised valdkonnad nagu näiteks audit, mis nõuab kõrgemal tasemel otsustus- ja analüüsivõimet, on osutunud keeruliseks automatiseerida.

Arvestuse infosüsteemide automatiseerimine tõstab arvestuse efektiivsust vähendades samas tööjõu osakaalu. Üleüldine automatiseerimise kiirus sõltub erinevatest teguritest. Peamised tegurid automatiseerimise kiirendamiseks on järgmised (Wilkinson 1991: 5):

- klientide surve vähendada kulusid,
- efektiivsuse eesmärkide täitmine,
- uute teenuste tutvustamine.

Kõik need tegurid suunitlevad arendama tarkvarasid arvestuse infosüsteemide automatiseerimiseks. Samas on automatiseerimist pidurdavaid tegureid. Üks neist olgu töökeskkond. Mõned ülesanded on kergemini automatiseeritavad kui teised. Väikese ettevõtte palgaarvestuse saab suuresti automatiseerida vastava tarkvara abil. Ettevõtte auditi koostamist on raskem automatiseerida tarkvara abil. Teine automatiseerimist pidurdav tegur on ettevõtte suurus. Suuremad ettevõtted omavad rohkem vahendeid infotehnoloogiasse investeerimiseks ning oskusi arendamaks tarkvara automatiseerimaks arvestuse infosüsteeme.

Arvestuse infosüsteemid loovad ettevõttele väärtust ja moodustavad olulise osa väärtusahelas. Arvestus muudetakse kiiremaks, usaldusväärsemaks ja modulaarsemaks. Arvestust automatiseeriva tarkvara arendamine võib oluliselt mõjutada efektiivsust,

millega arvestust läbi viiakse. Arvestuse infosüsteemile on ettevõttes omased järgmised funktsioonid (Accounting … 2016):

- koguda ja salvestada andmeid tegevuste ja ressursside ja pakkujate kohta,
- töödelda kogutud andmed juhatuse poolt otsustamisel kasutatavaks informatsiooniks,
- pakkuda kontrolli ressursside ja salvestatud andmete üle tagamaks nende kättesaadavuse.

Ettevõtetel on konkurentsis püsimiseks vaja pidevalt leida lahendusi tööoperatsioonide lihtsamaks ja kiiremaks muutmiseks. Kõik, mis otstarbekas, tuleb automatiseerida. Kuna tööjõukulud on reeglina üks ettevõtete suuremaid kuluallikaid, siis ei tasuks alahinnata tööjõu ajakulu mõõtmist ning selle automatiseerimisest tulenevat kasu. (Tööajaarvestuse … 2016) Sama kehtib palgaarvestuse kohta.

Töö- ja palgaarvestuse automatiseerimine võimaldab tööandjal töödelda töötajate andmeid arvutisüsteemi abil. Manuaalne töö- ja palgaarvestussüsteem eeldab, et töö- ja palgaarvestus tehakse käsitsi ja hõlmab seetõttu oluliselt aeglasemat protseduuri kui automatiseeritud süsteem. Automatiseerimine muudab töö- ja palgaarvestuse lihtsamaks ning vähendab vigade arvu, mis on rohkem omane manuaalsele süsteemile. Järgnevalt toob välja autor mõningad põhjused töö- ja palgaarvestuse automatiseerimiseks. Põhjused töö- ja palgaarvestuse automatiseerimiseks (Benefits ... 2016):

- tööarvestuse andmete transporditavus - andmed salvestatakse automaatselt andmebaasi, kust on neid võimalik mugavalt eksportida/importida palgaarvestusel kasutamiseks;
- maksude arvestamine - palgaarvestuse puhul on vaja arvestada erinevaid makse ning muid kitsendusi töötaja palga osas, mis on kiiresti tehtav kui protsessid on automatiseeritud;
- arvepidamine - töö- ja palgaarvestuse andmete kohta saab väljavõtteid teha aastate tagusesse aega.

Töö- ja palgaarvestuse automatiseerimine eeldab vastava tarkvara väljatöötamist. Järgnevalt on välja toodud funktsionaalsused, mis peaksid olema olemas töö- ja palgaarvestust automatiseerivas tarkvaras (Payroll … 2016):

- tööarvestus;
    - peab võimaldama arvestada töötunde töötajate, projektide, klientide, ülesannete ja materjalide kaupa;
    - ületundide arvestamine;
    - raportid töötatud tundide kohta;
    - raportid tasu kohta;
    - puhkuste ja haiguslehtede arvestus;
    - andmete eksporditavas;
    - ligipääs eri platvormidelt (mobiil, veeb, jne);
- palgaarvestus;
    - raportid arvestatud palkade, maksude, kompensatsioonide kohta;
    - kompensatsioonide ja ületundide jälgimine ja haldamine;
    - palgalipiku genereerimine;
    - TSD (tulu- ja sotsiaalmaksu, kohustusliku kogumispensionimakse ja töötuskindlustusmakse deklaratsioon) koostamine.

Kokkuvõttes on arvestuse infosüsteemid struktuurid, mille põhjal arvestusinfot salvestatakse, töödeldakse ja raporteeritakse. Tänapäeval arendatakse tarkvarasid nende protsesside automatiseerimiseks. Antud tarkvarade arendamiseks tuleb tunda ärindust, ärinduse protsesse, majandusarvestust ja tehnoloogiaid. Tarkvara kontseptsioon julgustab integratsiooni poole püüdlema (minimeerida andmete topelt töötlemist). Andmed on faktid, mis kogutakse, salvestatakse, hoiustatakse ja töödeldakse süsteemi poolt. Organisatsioon kogub andmeid sündmuste, mõjutatud ressursside ja sündmusega seotud osapoolte kohta. Informatsioon on andmetest erinev – informatsiooni moodustavad andmed, mis on töödeldud pakkumaks tähendust kasutajale. Tarkvara arendamine arvestuse infosüsteeme iseloomustavate protsesside automatiseerimiseks eelkõige muudab arvestuse efektiivsemaks ja vähem kulukaks.

1.2. Tarkvaraarenduse meetodid ja tööetapid

Järgnevalt kirjeldab autor töö- ja palgaarvestustarkvara loomise, hooldamise ja arendamise teoreetilisi tagamaid. Lähtutud on tarkvaratehnika üldistest ja enam levinud printsiipidest tarkvara loomisel. Teoorias on defineeritud etapid, mis tuleb läbida tarkvara loomise ja edasise arendamise käigus korduvalt. Tarkvaraarenduse tööetapid on järgmised (Sommerville 2011: 31):

1) nõuete väljaselgitamine,
2) kavandamine,
3) teostamine,
4) testimine,
5) tugi.

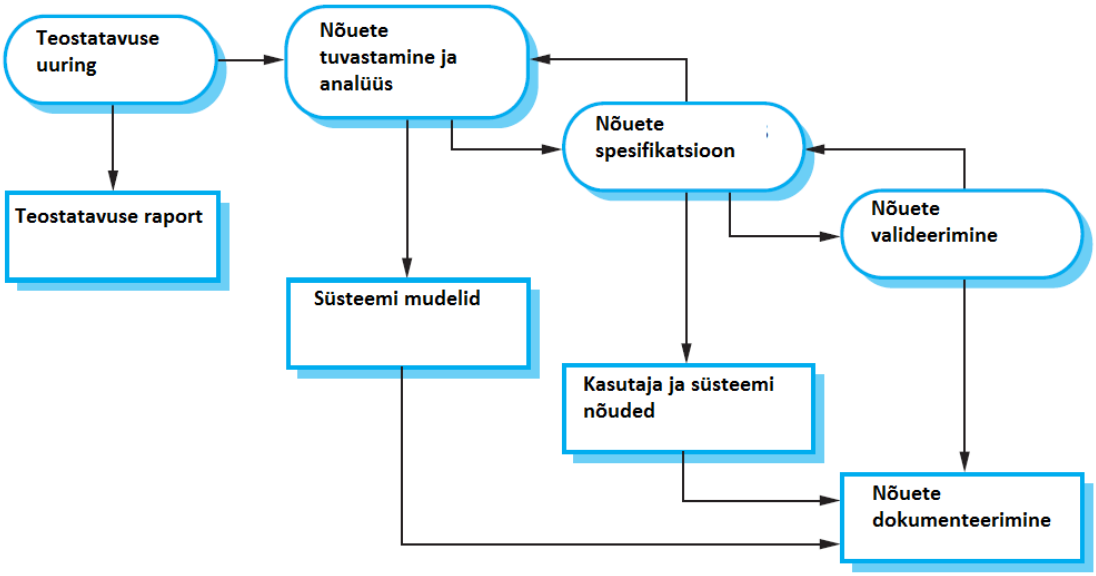
Enne tarkvara arendamist on vaja mõista keskkonda, milles tarkvara kasutama hakatakse. See hõlmab endas nii üldisemat tarkvarale esitatavate nõuete väljaselgitamist kui ka tarkvara kavandamist kõrgemal tasemel. Tuleb määrata riistvara, tarkvara, inimesed, andmestikud, protseduurid ja teised süsteemi elemendid ning tuvastada tegevused, mida süsteemis teha tuleb - analüüsida, spetsifitseerida, modelleerida, valideerida ja hooldada kõiki nõudeid. Selleks tuleb töötada koos kliendi, tulevaste kasutajate ja teiste võtmeisikutega. Enne ei saa tarkvara arendamist alustada, kui pole selge, millises ümbruses tarkvara funktsioneerima peab.

Esimeseks tööetapiks tarkvara arendamisel on nõuete väljaselgitamine. Ilma selleta ei ole võimalik järgmisi samme astuda. Nõuded tarkvarale klassifitseeritakse järgnevalt (Sommerville 2011: 321):

- funktsionaalsed nõuded - teenused, mida süsteem osutab ning kuidas käitub teatud situatsioonis;
- mittefunktsionaalsed nõuded - piirangud teenustes ja funktsionaalsuses (näiteks aeg).

Nõuete üleskirjutamiseks kasutatakse standardeid ning eristatakse selgelt vajalikke ja soovitatavaid nõudeid. Nõuete ülesmärkimise protsess on ülevaatlikult toodud joonisel 1. Joonisel toodud protsessis on neli peamist tegevust. Esiteks teostatavuse uuring, mille

raames tuvastatakse kas olemasolevate tehnoloogiatega on võimalik kliendi nõudeid täita. Teiseks nõuete tuvastamine ja analüüs, mille käigus tuvastatakse nõudeid potentsiaalsete klientidega arutades ning analüüsitakse nõudeid olemasolevate tarkvarade põhjal. Kolmandaks nõuete spetsifikatsioon, mille käigus dokumenteeritakse tuvastatud ja analüüsitud nõuded. Neljandaks nõuete valideerimine, mille abil tuvastatakse vead dokumenteeritud nõuetes ning parandatakse need. (Sommerville 2011: 37)



Joonis 1. Nõuete väljaselgitamise tööetapi protsess (Sommerville 2011: 38)

Nõuete uurimiseks on Pressmann toonud välja järgmised tehnikad (Pressmann 2015: 143):

- intervjuu klientidega;
- stsenaariumid - pikemad protseduuride kirjeldused kuidas organisatsioonis seda või teist toimingut läbi viiakse;
- prototüüpimine - luuakse tarkvara mudel, millest saab edasi arendada projekti.

Nõuete leidmise ja tarkvara kavandamise vahele jääb kasutusmallide kokkupanemine. Samuti stsenaariumide koostamine, kuidas süsteemi töötegemisel kasutatakse. Leitakse erinevat tüüpi inimesed või seadmed, mis süsteemiga suhtlema hakkavad.

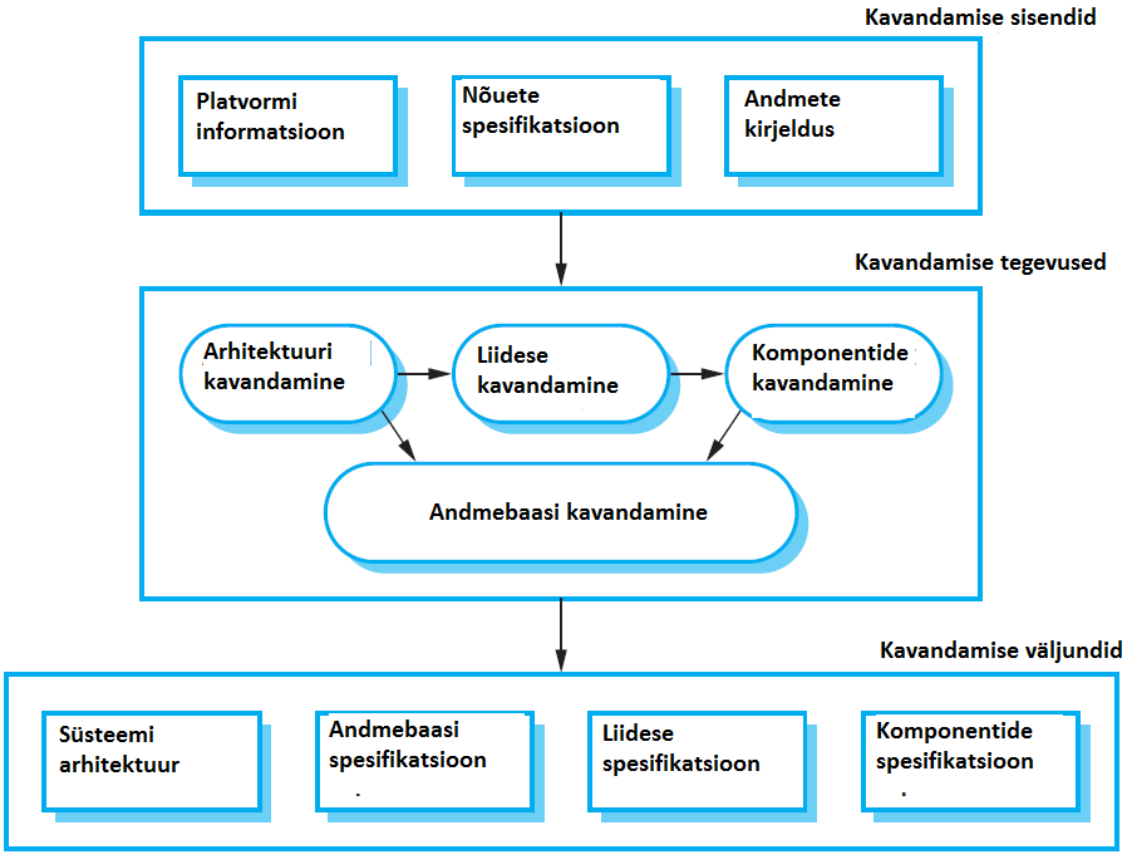
Tulemuste põhjal tehakse andmemudel, infoliikumise ja juhtimise mudel ning käitumise mudel. Määratakse tarkvara liides teiste süsteemi osadega ning piirangud, millele tarkvara peab vastama. Tarkvara kavandajale antakse ette info, mille saab muuta andmekavandiks, arhitektuuri kavandiks, liideste kavandiks ja komponentide kavandiks. Tekkinud mudelite alusel peab olema võimalik hinnata valmis tarkvara kvaliteeti ennekõike tagatava funktsionaalsuse aspektist. (Pressmann 2015: 150)

Järgmiseks tööetapiks on tarkvara disain (kavandamine). Vastavalt IEEE (Elektri- ja Elektroonikainseneride Instituut) definitsioonile kujutab see endas süsteemi või komponentide arhitektuuri, osade, liideste ja teiste omaduste määramist. Tarkvara elutsüklis on kavandmine protsess, kus analüüsitakse nõudmisi loomaks tarkvara sisemise struktuuri ja organisatsiooni kirjeldus. Loodud kirjeldus on omakorda realisatsiooni aluseks. Tarkvara projekt peab kirjeldama kuidas süsteem on jaotatud komponentideks ning millised on komponentide liidesed. Komponendid peavad olema kirjeldatud sellise täpsusega, mis lubaks hakata neid realiseerima. Klassikalises tarkvara elutsüklis on kavandamise osa jaotatud kahte etapp (Sommerville 2011: 38):

- arhitektuuri kavandamine, millega määratakse kõrgemal tasemel kindlaks komponendid, seosed suuremate ja üldisemate tarkvara komponentide vahel;
- detailsem kavandamine, millega täpsustatakse komponentide ülesehitus (protseduurid, objektid, jne).

Tekkinud kavandit on reeglina võimalik analüüsida veendumaks tarkvarale esitatud nõuete täitmises. Võib ka tekkida mitu kavandit, mida võrrelda saab. Korralik kavand peab tagama arendatava tarkvara kvaliteedi. Kavandi järgi peab saama hinnata tarkvara kvaliteeti. Kavand on aluseks kõigile järgmistele sammudele. Kavandi loomise protsessist annab hea ülevaate joonis 2. Joonisel on toodud välja neli olulist tegevust kavandi loomise protsessist. Arhitektuuri kavandamine hõlmab endas tarkvara üldise struktuuri (komponendid ja nende integratsioonid) kavandamist. Liidese kavandamine kujutab endast komponentide integratsioonide loomist teadmata kuidas üksikud komponendid töötavad. Komponentide kavandamine hõlmab endas tarkvara üksikute komponentide funktsionaalsuste kavandamist. Andmebaasi kavandamine kujutab endast tarkvara andmestruktuuri loomist. (Sommerville 2011: 40) Eelnevate kavandamise

tegevuste väljundina luuakse tarkvara kui süsteemi üldine arhitektuur ning erinevate komponentide spetsifikatsioonid.



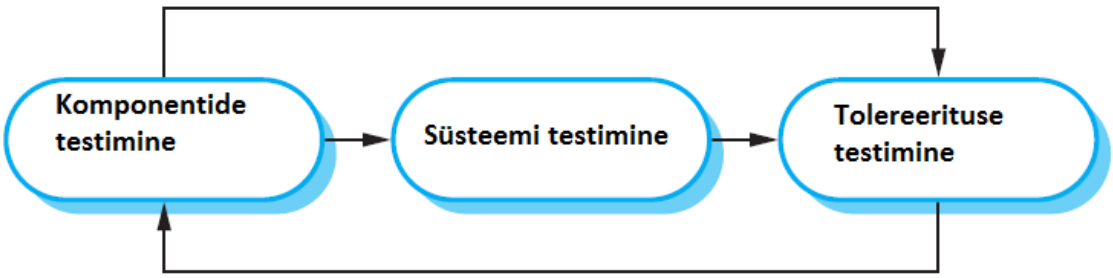
Joonis 2. Kavandi loomise protsess (Sommerville 2011: 39)

Kavandi valmimisel saab liikuda edasi realisatsioonifaasi. Realisatsioonifaas on ilmselt kõige olulisem, sest sellega arendatakse valmis tarkvara ise. Realisatsioonifaas järgneb kavandamisele ja tihti on raske nende kahe vahele selget joont tõmmata. Kui kavandamine ja peenendamine on jõudnud nii kaugele, et üksik arendaja võib ühe osa kallal tööle asuda, saab öelda, et on jõutud realisatsiooni faasi. Väiksem tarkvara, kus mõõduks ongi paras korraga realiseeritav tükk, ei pruugi nõuda eraldiseisvat kavandamisetappi. Samal ajal suured projektid võivad nõuda mitmeid iteratsioone kavandamise ja realisatsiooni vahel, kui luuakse rohkem prototüüpe ja neid kasutatakse või kõrvale heidetakse. Realisatsioonifaas hõlmab endas tarkvara arendamist ning millise see välja näeb sõltub kasutatavast tarkvaraarenduse metoodikast. (Pressmann 2015: 64)

Kui lähtekood on valmis, tuleb tarkvara testida, et avastada ja kõrvaldada nii palju vigu kui vähegi võimalik enne tarkvara kliendile üle andmist. Tarkvara arendamine jõuab testimise tööetappi. Tuleb luua mitmeid testjuhte, mis võimalikult suure tõenäosusega vigu avastaksid. Vigu on vaja leida sisemises loogikas ja sisendis/väljundis (käitumises, jõudluses, funktsionaalsuses). Testimise teooria on vajalik leidmaks õiged ja vajalikud testjuhtumid, mis süstemaatiliselt vigu otsiksid (ja leiaksid). Toimub tarkvara kvaliteedi hindamine, vigade ja probleemide identifitseerimine. Testimisega ei saa tõestada, et tarkvaras vigu ei ole. Seetõttu on testimisel tihe seos hooldamise tööetapiga. Testimise eesmärgid (Sommerville 2011: 40):

- testimine on tarkvara käivitamise protsess eesmärgiga leida vigu,
- hea test leiab suure tõenäosusega veel leidmata vea,
- edukas test leiab veel leidmata vea.

Testimine tänapäevases mõistes peab saatma kogu arendusprotsessi. Ülevaate testimise etappidest annab joonis 3. Testimine koosneb kolmest etapist. Alustuseks testitakse tarkvara üksikuid komponente. Järgmisena testitakse tarkvara üksikute komponentide integratsioone ja koostööd. Viimaks testitakse tarkvara kliendi andmetega.



Joonis 3. Testimise protsessi etapid (Sommerville 2011: 41)

Peale tarkvara realisatsioonifaasi leiab tarkvara esimese kliendi või toimub tarkvara uuendamine olemasolevate klientide poolt. Järgneb hooldamisfaas. Tarkvara hooldamiseks nimetatakse tarkvara muutmist peale kliendile üleandmist parandamaks vigu, jõudlust või muid omadusi.

Hoolduse tööetapi eesmärk on hoida tarkvara töös nii kaua kui võimalik. Ajalooliselt on sellele tööetapile vähem tähelepanu pööratud. Teravalt kerkis hooldamise probleem esile mitte väga ammu. Seoses uue aastatuhande tulekuga. Hooldamise juures on oluliseks aspektiks teiste arendajate poolt kirjutatud tarkvaraga töötamine (parandamine, täiendamine, muutmine). Hooldusele võib saada tuge avatud lähtekoodiga tarkvara maailmast, sest seal kasutatakse palju teiste kirjutatud tarkvarasid. (Pressmann 2015: 796)

Tarkvara tootmine lõppeb kliendile tarkvara üleandmisega. Valminud tarkvara peab olema selline, mida klient tahtis. Lisaks peab tarkvara edasi arenema. Tarkvara kasutamise käigus leitakse anomaaliaid, muutub töö keskkond, tulevad uued nõuded. Muudatuste vajadused logitakse, määratakse muudatuste mõju, tarkvara koodi muudetakse, tehakse testid, antakse välja tarkvara uus versioon ja vajadusel luuakse ka õpetus ja dokumentatsioon. (Pressmann 2015: 814)

Järgnevalt kirjeldab autor levinumaid tarkvaraarenduse meetodeid. Tarkvaraarenduse meetodid jagatakse järgmisel (Awad 2005: 2):

- traditsioonilised tarkvaraarendusmeetodid,
- agiilsed tarkvaraarendusmeetodid.

Traditsioonilise meetodi arenduslaade iseloomustab hoolikas projektiplaneerimine, formaliseeritud kvaliteeditagamine, arvutipõhiste tarkvaratehnikate kasutamine analüüside ja projekteerimismeetodite jaoks ning range ja juhitud tarkvara arendamise protsess. Klient näeb valminud tarkvara alles lõppfaasis. Seetõttu võib arendaja dokumentatsioonist aru saada selle kirjapanijast erinevalt. Projekti alguses soovitakse kindlat tarkvara, kuid projekti käigus soovid muutuvad. Traditsiooniliste tarkvaraarendusmeetodite korral on neid muudatusi keeruline rakendada ja tarkvara valmimise aeg pikeneb. Uuenduste tegemisel tuleb muuta dokumentatsiooni ja arendamisele eelnevalt tuleb läbida põhjalik analüüs. (Ghilic-Micu 2013: 64)

Agiilsed tarkvaraarenduse meetodid tähtsustavad kavandamise ja teostamise tööetappe. Lisaks kaasatakse tegevusi nagu näiteks nõuete väljaselgitamine ning testimine tarkvara kavandamisel ja juurutamisel (Sommerville 2011: 59). Nõuded jagatakse

kasutusmallideks ja kasutusmalle hakatakse ükshaaval teostama. See võimaldab arendamise käigus uuendusi sisse tuua, sest igal tarkvara osal on omad nõuded.

Agiilsed tarkvaraarenduse meetodid erinevad traditsioonilistest mitmetes aspektides. Paindlike meetodite puhul on meeskond väike ja ei teki bürokraatiat – oluline on koostöö ja suhtlus. Klienti võib käsitleda meeskonna liikmena. Klient hoitakse projekti käigus kursis arendatavaga. Nõuded võivad muutuda ja kliendi kaasamine väldib lõppfaasis tarkvara erinemist kliendi poolt soovitust.

Traditsiooniliste meetodite puhul järgitakse kindlat mudelit. Ühe projektiga tegelevad mitu erinevat meeskonda. Sellest tulenevalt peavad meeskonnad olema suured ja projekti juhtimine range. Tarkvara nõuded on projekti alguses määratletud ning ei muutu.

Agiilseid tarkvaraarendusmeetodeid eelistatakse väiksemate ning keskmise suurusega projektide korral. Rõhutakse tarkvara kiirele ja odavale valmimisele. Traditsioonilist tarkvaraarendusmeetodit kasutatakse suurte projektide korral. Rõhutakse kõrgele turvalisusele. (Awad 2005: 26)

Tänapäeval kasutatakse aina rohkem agiilseid tarkvaraarendusmeetodeid, sest aina keerulisem on ette ennustada arenduse käigus tulevaid probleeme (Awad 2005: 3). Muudatust nõudvat probleemi on lihtsam lahendada agiilse tarkvaraarendusmeetodi abil. Lahendust saab kohe otsima hakata väike projektiga tegelev meeskond. Traditsioonilise tarkvaraarendusmeetodi puhul on muutuste arendamisele eelnevalt vaja erinevatel meeskondadel muutusi analüüsida ja disainida.

Järgnevalt tutvustab autor levinumaid agiilseid tarkvaraarendusmeetodeid. Azizyan poolt korraldatud küsitluse põhjal 54% vastanutest kasutavad SCRUM tarkvaraarendusmeetodit, 32% vastanutest arendavad tarkvara kasutades SCRUMi koos ekstreemprogrammeerimisega (edaspidi XP), puhast XP-d kasutavad 11% (Azizyan 2011: 34). Teisi agiilseid tarkvaraarendusmeetodeid kasutavad kokku alla kümne protsendi vastanutest – nende seas Crystal. Järgnevalt kirjeldab autor järgmisi levinumaid agiilseid tarkvaraarendusmeetodeid:

- SCRUM,
- ekstreemprogrammeerimine (XP),

- Crystal.

SCRUM on üks populaarsemaid agiilseid tarkvaraarendusmeetodeid. Antud tarkvaraarendusmeetodi eesmärk on aidata väikestel meeskondadel arendada keerulisi tarkvarasid (Louise 2012: 5). Eesmärk on hoida komplitseeritud ärikeskkonda lihtsana.

SCRUMi puhul pole projekti eelarve enne projekti paigas. Projekti arendamise maksumus selgub projekti lõppfaasis. Fookus on arendusmeeskondade toimimisel, loomaks paindlikku ja produktiivset süsteemi muutuvas keskkonnas (Ghilic-Micu 2013: 74).

SCRUM põhineb meeskonna suhtelisel iseseisvusel ja kohanemisvõimel. Meeskonna suhteline iseseisvus seisneb selles, et projekti omanik annab meeskonnale ülesanded, kuid meeskond ise otsustab ülesannete lahendamise käigu suurendamaks tootlikkust. Spetsiifilist tarkvaraarengu tehnikat ei nõuta, kuid meeskond peab järgima meetodi põhiskeemi vältimaks segaduse tekkimist. Segadus võib tuleneda projekti keerukusest ja ettearvamatusest. (Ghilic-Micu 2013: 65)

SCRUMi iseloomustavad sprindid, mis on kahe kuni nelja nädalase kestvusega perioodid tarkvara osa valmis arendamiseks ja testimiseks. Ühe sprindi jooksul realiseeritavad nõuded on kirja pandud tarkvara tööde nimekirjas, kus on kõik tarkvara nõuded järjestatud tähtsuse järgi. Sprindi planeerimise koosolekul otsustatakse sprindi jooksul täidetavad nõuded. Tarkvara omanik paneb paika tarkvara osad, mida meeskond peab arendama sprindi jooksul, ning meeskond otsustab kui suure osa soovitud nõuetest saab täita. Sprindi ajal on keelatud tööde nimekirja muuta. Iga päev on koosolek arutamaks eelneval päeval tehtut ning eelseisva päeva plaani. Sprindil on kindel tähtaeg. Nõuded, mida ei täideta tähtajaks, jäetakse teostamata ning alustatakse uue sprindi planeerimist. (Glossary … 2016) Sprindi lõpus arutatakse projekti tellijaga valminud tarkvara osa ja määratakse muutused, mida järgmise sprindi lõpuks esitleda tuleb (Rossberg 2008: 70).

XP toetab tarkvaraarendust väikestes meeskondades ning aitab toime tulla ebamääraste ja muutuvate nõuetega. Eelnimetatud agiilne tarkvaraarendusmeetod vastandub paljuski traditsioonilisele tarkvaraarenduse eeldustele. Näiteks XP põhjal pole valmis tarkvara ühe osa muutmine kallis ning keeruline (Beck 1999: 56). XP algatajate eesmärk oli luua

tarkvaraarendusmeetod projektide arendamiseks kuni kümneliikmelistes meeskondades (Williams 2007: 214). Antud tarkvaraarendusmeetod põhineb neljal väärtusel: suhtlus, lihtsus, tagasiside ja julgus (Hunt 2006: 4).

Suhtluse väärtust kujutab endast suhtlusele keskenduvate tegevuste praktiseerimine. Tegevused nagu ühiktestimine ja paarisprogrammeerimine on mõistlikud isegi lühiajalisel kasutamisel. Eeltoodud tegevuste mõjul peavad arendajad, kliendid ning juhid omavahel suhtlema (Beck 1999: 54). Kommunikatsiooni väärtus põhineb tähelepanekul, et projekti probleemid tekivad tihti olematu suhtluse tõttu (Williams 2007: 43).

Lihtsuse väärtus kujutab endast lihtsaimat kliendi nõuetele vastavat versiooni tarkvarast. Disainida ja kodeerida tuleb ainult kliendi poolt nõutut (Williams 2007: 44). Mida lihtsam on tarkvara kood, seda kergem on leida koodist vigu ja neid parandada. Lahendus ise ei pea olema lihtne, aga peab olema lihtsaim lahendus probleemile. (Hunt 2005: 178).

Tagasiside väärtus on oluline mitmest aspektist. Arendaja kirjutab ühikteste iga tarkvara osa kohta, mis võib katki minna. Tarkvarasse uue tüki lisamisel jooksutatakse kõiki teste veendumaks vigade puudumises. Arendajad saavad tagasisidet juhatajalt, kes jälgib töö arengut. Oluline on kõigi teatud ajaks määratud eesmärkide täidetus (Beck 1999: 3). Arendusmeeskond saab tagasisidet kliendilt pärast iga uut tarkvara versiooni. Klient teavitab soovitavatest muutustest (Williams 2007: 214).

Kolm eelnimetatud väärtust viivad neljandani ehk lubavad arendusmeeskonnal olla julged. Julgust on vaja muutmaks tarkvara paremaks ilma uusi vigu tekitamata. (Hunt 2006: 43).

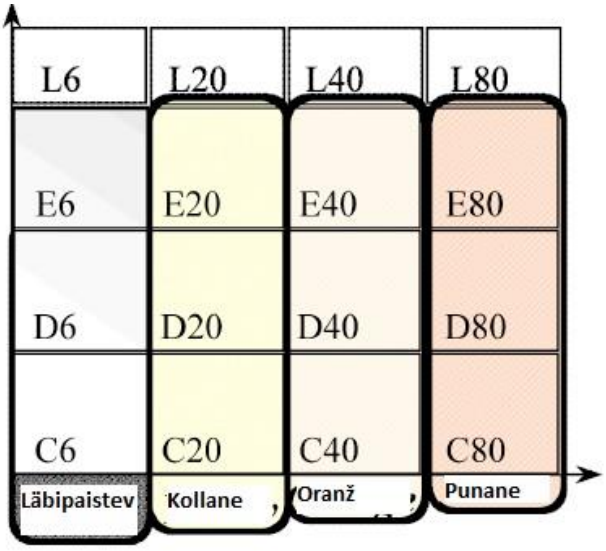
Võttes arvesse nelja eelnimetatud väärtust on koostatud kaksteist tava aitamaks neid väärtusi täita (Hunt 2005: 180):

- plaanimismäng – keskendub järgmise väljalaske planeerimisele;
- väikesed väljalasked – tarkvarasüsteemi arendatakse väikeste väljalasetena lisades iga korraga süsteemi funktsioone;
- lihtne disain – koodi hoitakse võimalikult lihtsana;
- testimine – ühikteste tuleb pidevalt täiendada ja arendamise jätkamiseks peab kood läbima testid;

- ümberstruktureerimine – süsteemi täiustamine funktsionaalsust muutmata;
- paarisprogrammeerimine – kood arendatakse paaris sama arvuti taga töötavate arendajate poolt;
- kollektiivne omand – kogu kood kuulub kõigile ning igaühel on õigus seda parendada;
- pidev integreerimine – uus kood on integreeritud ning süsteem ehitatakse uuesti üles iga kord kui mõni ülesanne on täidetud;
- meeskonnatöö – klient on osa meeskonnast ning alati valmis vastama küsimustele;
- ühtne kodeerimisstandard – kommentaare, meetodeid ja muutujaid tähistatakse samas stiilis;
- 40-tunnine töönädal – arendajad on alati värsked ning valmis väljakutsetele;
- süsteemi metafoor – süsteemi ehitust kirjeldatakse lihtsa metafooriga, millest kõik aru saavad.

Crystal on agiilsete tarkvaraarendusmeetodite perekond. Antud tarkvaraarendusmeetodite puhul ei arendata tarkvara korraga ühes tükis, vaid väikeste osade kaupa. Rõhku pannakse inimeste tihedale suhtlusele. Puudub üks kindel Crystal meetod. Erinevate projektide jaoks on erinevad Crystal metoodikad (Cockburn 2004: 17). Eristatakse nelja erinevat agiilset tarkvaraarendusmetoodikat. Kõik on samade põhimõtetega, aga kasutatavus sõltub projekti suurusest ja raskusest (Cockburn 2006: 15). Kõige agiilsem versioon on läbipaistev kristall, millele järgnevad kollane kristall, oranž kristall ja punane kristall (Williams 2007: 215).

Joonis 4 annab ülevaate Crystal tarkvaraarendusmeetoditest. X-telg määrab meeskonna suuruse. Mida suurem meeskond, seda raskem on juhtida protsesse omavahelise suhtlemisega ja seda vajalikum on dokumentatsioon. Suurema meeskonna juhtimiseks on vaja sobilikku tarkvaraarendusmeetodit. Y-telg iseloomustab projekti keerukust. Ülespoole liikudes muutub tegevus olulisemaks ja keerulisemaks. Joonisel on tähistatud mugavus (C), vaba raha (D), hädavajalik raha (E) ning kriitilisus (L). Kui projekt pole kriitilise tähtsusega ning selle täitmiseks pole vaja suurt meeskonda, siis tasub seda arendada läbipaistva tarkvaraarendusmeetodiga. Kui projekt on keeruline ja kallis, siis sobib arendamine suurema meeskonna ja punase tarkvaraarendusmeetodiga (Cockburn 2006: 113).



Joonis 4. Värvide järgi järjestatud Crystal meetodid (Cockburn 2004: 240)

Kõik Crystal tarkvaraarendusmeetodid tähtustavad arendusmeeskonda. Arendusprotsess on endiselt oluline, kuid sekundaarne. (Williams 2007: 216). Crystal metoodikad põhinevad järgneval seitsmel väärtusel (Cockburn 2004: 33):

- pidevad väljalasked – testitud kood esitletakse kliendile pärast iga tarkvara osa valmimist;
- tagasisidestatud areng – meeskond peab koosolekuid tutvustamaks hetkeseisu;
- tingitud suhtlemine – inimesed paigutatakse ühte tööruumi hoidmaks kõik jooksvalt probleemidest kursis;
- isiklik turvalisus – võimalus rääkida enda probleemidest teiste pahameelt kartmata;
- fookus – projektiga keskendunult tegelemiseks tuleb mõista klienti ehk millega klient tegeleb ja miks tal seda tarkvara vaja on;
- kogenenud kasutajate kättesaadavus – kogenud kasutajad annavad kvaliteetset tagasisidet puuduste ja soovide kohta;
- tehniline keskkond - tagatud peab olema automatiseeritud testimine, konfiguratsioonihaldus ning pidev integratsioon.

Töö- ja palgaarvestustarkvara arendamise protsessi läbitegemine on omane töö- ja palgaarvestuse automatiseerimisele ning võimaldab muuta manuaalselt läbitud protsessid

automaatseteks. Arendamise käigus kasutatakse tarkvaratehnikas levinud tööetappe ning meetodeid.

1.3. Töö- ja palgaarvestustarkvarade arendamist mõjutavad tegurid

Töö- ja palgaarvestustarkvarade arendamisel on olulised konkurentsis püsimine ja koostöö klientidega. Eelnev tagab töö- ja palgaarvestustarkvara arendavale ettevõttele ressursid tarkvara arendamise jätkamiseks tulevikus. Kliendid näevad tihti erinevaid takistusi töö- ja palgaarvestustarkvara juurutamisega. Töö- ja palgaarvestustarkvara üldine eesmärk on koguda, organiseerida ja raporteerida ettevõtte tehingute ja sündmuste andmeid. Nende andmete põhjal saab genereerida erinevaid raporteid ettevõtte juhtkonna abistamiseks otsuste langetamiseks. Kliendi või kliendigrupi jaoks sobiva tarkvara arendamiseks tuleb arvestada mitmete teguritega. Järgnevalt toob autor välja töö- ja palgaarvestustarkvara arendamist mõjutavate teguritena motivaatori, takisti ja kiirendi.

Motivaator motiveerib töö- ja palgaarvestustarkvara arendavat ettevõtet tarkvara arendama. Antud juhul käsitleb autor motivaatorina konkurentsieelist. Erinevad töö- ja palgaarvestustarkvarad eksisteerivad tihti tugeva konkurentsiga turul. Sellest tulenevalt peavad tarkvara arendavad ettevõtted omama eelist konkurentide ees. Konkurentsieelis on võtmeküsimus strateegilises juhtimises andes eelise. Eelis võib olla organisatsiooni võimetes - organisatsioon teeb midagi, mida teised ei tee või teevad halvemini. Samas võib konkurentsieelis esile kerkida organisatsiooni varadest või ressurssidest - organisatsioonil on midagi, mida konkurentidel ei ole.

Konkureerimise vorm ja konkurentsieelise kujundamine on üheks põhiliseks teguriks ettevõtte säilimise tagamisel. Michael Porteri (Porter 1991: 96) käsitluses on konkurents ettevõtte edu või ebaedu algpõhjuseks. Oluline on ettevõtte konkurentsisituatsiooni määramine turul ja vastavate strateegiate kujundamine. Porter on välja töötanud erinevaid teooriaid konkurentsi käsitlemiseks. Neist tuntuim on Porteri viie konkurentsijõu mudel, mille kohaselt määravad konkurentsiolukorra tegevusharus viis jõudu (Porter 1991: 100):

- Uute sisenejate oht ja sisenemisbarjäärid. Mastaabisääst, margitoote lojaalsus ja kapitalivajadused määravad uue konkurendi tegevusalasse sisenemise raskuse või kerguse.
- Olemasolev konkurents. Nõudluse suurenemine või vähenemine ja toote erinevused määravad konkurentsivõistluse tegevusala ettevõtete hulgas.
- Ostjate võim. Ostjate arv, informeeritus ja asendustoodete kättesaadavus määravad ostjate mõju tööstusharus. Suurem arv ostjaid tagab väiksema võimu.
- Tarnijate võim. Varustajate kontsentratsioon ja asendussisendite kättesaadavus määravad varustajate võimu.
- Asendajate oht. Tegevusalal eksisteeriv asendustoodete oht ehk kui palju neid on ja kui arvestatavad need tooted on.

Põhimõte on lihtne – eeltoodud tegurite mõjude tugevnemisel väheneb ettevõtte kasum. Ideaalses situatsioonis on kõik konkurentsijõud nõrgad, kuid praktikas seda ei esine.

Porteri põhjal on püsivat konkurentsieelist võimalik saada eristumisega, mida konkurendid järele ei suuda teha. Töö- ja palgaarvestustarkvara arendavad ettevõtted peavad leidma funktsionaalsusi, mida konkurendid ei arenda ning kliendid soovivad. Põhipunkt Porteri käsitlusest on see, et ettevõtte edukuse tagab jätkuv konkurentsieelis. Kaks põhimõttelist konkurentsieelise loomise moodust (Porter 1991: 105):

- Tegutsemisefektiivsuse läbi. Teha sama asja, mis konkurendid, kuid teha seda paremini ehk efektiivsemalt, kiiremini, parema kvaliteediga, väiksema kuluga.
- Strateegilise positsioneerumise läbi. Teha midagi sellist, mida konkurendid ei tee ehk pakkuda kliendile midagi unikaalset.

Porter kritiseerib eesmärgi seadmiseks ainult kiirust ja paindlikkust. Kiirus ja paindlikkus ilma kindla strateegilise positsioonita ei taga lisakasumit. Töö- ja palgaarvestustarkvara tootvad ettevõtted omavad konkurentsieelist tagamaks motivatsiooni tarkvara edasi arendada.

Ettevõtluskliima on muutunud aasta-aastalt keerulisemaks ning turud muutuvad aina globaalsemaks. Juhatus on pideva surve all säilitamaks konkurentsivõimet vähendades ettevõtte kulusid ning muutes erinevaid protsesse efektiivsemaks – sealjuures on tähtsal

kohal töö- ja palgaarvestuse efektiivsemaks muutmine. Konkurentsis püsimiseks peab ettevõte käima kaasas tehnoloogiliste trendidega ja olema võimeline kohanema uute oludega, paljud seda aga ei ole ning see põhjustab ettevõttele erinevaid probleeme. Ettevõttest saab klient töö- ja palgaarvestustarkvara arendavale ettevõttele.

Töö- ja palgaarvestustarkvara arendav ettevõte peab tagama klientide olemasolu jätkamaks tarkvara arendamist. Klientidele tarkvara pakkumisega kaasneb tihti projekt tarkvara juurutamiseks kliendi juures. Sellega võivad tihti kaasneda takistused arendamisel kui kliendist räägitakse mööda või klient ei ole tarkvaraga rahul.

Töö- ja palgaarvestustarkvara juurutamine kliendi juures algab kliendi valmisoleku hindamisest. See on oluline töö- ja palgaarvestustarkvara arendamisel veendumaks, et tarkvara sihtturule antud tarkvara sobib. Järgnevalt on välja toodud viis peamist tegurit kliendi valmisoleku hindamiseks (Saremi *et al* 2007: 60):

- kultuuri tegur – kliendi ettevõtte sisekliima toetab tiimitööd ning on vastuvõtlik muutustele;
- kliendi ettevõtte võimsustegur – kliendi ettevõtte võime leida ressursid toetamaks töö- ja palgaarvestuse integreerimise projekti;
- toe tegur – kliendi ettevõtte juhtkonna toetus ja oskus anda juhtimine üle projektijuhile ja ekspertidele;
- motivatsiooni tegur – kliendi ettevõtte võime näha töö- ja palgaarvestuse integreerimise positiivseid külgi;
- IT tegur – IT spetsialistide olemasolek süsteemi integreerimiseks kliendi ettevõttes.

Kui töö- ja palgaarvestustarkvara arendava ettevõtte potentsiaalse kliendi valmisolek on hinnatud tuleb uurida põhjuseid tarkvara juurutamiseks. Põhjendamine aitab paremini mõista kliendi reaalseid vajadusi. Vinatoru ja Calota on oma teadustöös toonud välja viis peamist põhjust juurutamise projekti läbimiseks kliendi ettevõttes. Ühtlasi on need suureks takistuseks projekti elluviimisel. Põhjused juurutamise projekti läbimiseks kliendi ettevõttes on järgmised (Vinatoru *et al* 2014: 104):

- finantsandmete integreerimine – juhtkonnal ning töö- ja palgaarvestuse töötajatel on lihtsam andmeid erineva nurga alt analüüsida kui on tsentraalne andmebaas ja andmete organiseeritus;
- andmebaasi käskude integratsioon – süsteem võimaldab teha reaalajas muudatusi nagu näiteks mitme töötaja tööajad saab andmebaasi kanda reaalajas hetkega;
- protsesside standardiseerimine – ettevõtted kasutavad tihti arvestuses erinevaid meetodeid erinevate protsesside puhul ja see võib tihti tekitada segadust, kuid keskne süsteem võimaldab standardiseerida kasutatavad meetodid;
- protsesside optimiseerimine – süsteem võimaldab omada pidevat ülevaadet töö- ja palgaarvestuses eri nurkade alt ning sel viisil leida üles ebaefektiivsusi süsteemist;
- värbamisspetsialistidele kättesaadava informatsiooni standardiseerimine – tihti pole värbamisspetsialistidel reaalajas ülevaadet töötajate puhkustest, boonustest, hüvitistest ja tööaegadest, kuid süsteem võimaldab selle probleemi lahendada.

Töö- ja palgaarvestustarkvara arendamist takistavad ebaedukad tarkvara juurutamise projektid. Kliendipoolsel juurutamisel on oluline hinnata realistlikult kulusid, mis tuleb teha projekti vältel. Kulude realistlikuks hindamiseks tuleb teha koostööd töö- ja palgaarvestustarkvara arendavad ettevõttel ja kliendil. Ühtlasi tuleb omada ülevaadet kulude hindamisel suurimaid vigu tekitavate projektikomponentidega. Töö- ja palgaarvestustarkvara arendamisel ja kliendile pakkumisel tuleb luua kliendile realistlik ülevaade projektiga tulenevatest kuludest. Ühtlasi tuleb pakkuda tuge nii enne kui peale projekti algust. Järgnevalt on toodud välja kõnealuste projektidega seoses enim alahinnatud kulud (Vinatoru *et al* 2014: 108):

- koolitamine,
- integratsioon ja testimine,
- seadistamine,
- andmete import/eksport,
- andmete analüüs,
- konsultatsioonid.

Töö- ja palgaarvestustarkvarade arendamist kiirendab kliendibaasi kasv ning edukate juurutamiste läbiviimine. Tuginedes paremini tuntud tavadele projektide elluviimises, on töö- ja palgaarvestustarkvarasid arendavatel ettevõtetel võimalik edukaid projekte läbi viia minimaalsete kuludega. Erinevad autorid on kirjutanud teadusartikleid majandustarkvarade juurutamisest klientide ettevõtetes. Autor on erinevaid teadusartikleid analüüsinud ning võtnud töö- ja palgaarvestustarkvara juurutamise kokku järgmiselt nüansside loeteluna (Xu *et al* 2011: 293):

- süsteemi vajalikkuse põhjendamine;
- süsteemi valimine;
    - võetakse ühendust võimalikult paljude pakkujatega ning uuritakse erinevate süsteemide hindade/spetsifikatsioonide kohta;
    - leitakse ettevõtte vajadustega kõige paremini kattuv süsteem;
- projektiplaani koostamine;
- süsteemi implementeerimine;
    - uue süsteemi seadistamine;
    - vana süsteemi andmete analüüs, eksport ja import uude süsteemi;
    - töötajate koolitamine uut süsteemi kasutama;
- konsultatsioonid ja tugi.

Lõpetuseks saab tarkvara arendamist mõjutavate tegurite puhul välja tuua mitmete autorite poolt kooskõlastatud nimekirja viiest tegurist. Eelmainitud positiivse mõjuga tegurid on järgmised (McLeod *et al* 2011: 26):

- nõuete väljaselgitamine,
- projektijuhtimine,
- tarkvaraarendusmeetodi kasutus,
- kliendi kaasamine,
- kliendi koolitamine,
- muudatuste juhtimine.

Nõuete väljaselgitamine on kriitiline mõjutegur tarkvara arendamisel (Alvarez 2002: 101). Nõuete väljaselgitamine hõlmab kliendiga ühisele arusaamale jõudmist arendatava

tarkvara informatsiooni, protsesside ja funktsioonide osas. Eduka projekti jaoks on vajalikud selgelt kirja pandud nõuded (Lemon *et al* 2002: 30). Ebamääraselt kirjeldatud nõuetega tarkvaraprojektid tihti ebaõnnestuvad tulenevalt ressurssidega tehtud valearvestusest (Butler *et al* 1999: 355).

Projektijuhtimine hõlmab tarkvaraprojekti juhtimist ning ressursside organiseerimist ja juhtimist projekti käigus. Mitmed autorid on välja toonud tarkvaraprojekti edukuse sõltuvuse juhtimisprotseduuridest ja projektijuhi oskustest. (Johnson *et al* 2001: 15)

Tarkvaraarendusmeetodi poolt määratud protseduurid aitavad tarkvaraprojekti organiseeritult arendada. Kindla tarkvaraarendusmeetodi kasutus mõjutab projektist osavõtjate omavahelist suhtlust ja suhteid. See aitab parandada töökäiku tagades arendajate rahulolu. (Robey 2001: 89)

Kliendi kaasamine mõjub positiivselt tarkvaraprojekti edukusele ja kliendi hilisemale rahulolule (Coombs *et al*, 1999: 146). Kliendi kaasamata jätmist käsitletakse tõsise riskina tarkvaraprojektile. Kliendi kaasamine tarkvaraprojekti algusfaasis omab suuremat mõju kui kliendi kaasamine tarkvaraprojekti lõppfaasis (Foster *et al* 1999: 331). Lisaks võimaldab see tekitada grupi suhtluseks arendajate ja kliendi vahel. Selline suhtlus võimaldab kliendil selgelt oma huvisid, eesmärke ja nõudeid esitada. Samas võimaldatakse arendajatel kaasa rääkida. Kliendi mitte kaasamine arendamise protsessi võib viia kliendi nõuetele mittevastava tarkvara arendamiseni. Lisaks võib klient keelduda arendatud tarkvara kasutamast.

Kliendi koolitamine on oluline tarkvara arendamist mõjutav tegur. Erinevate autorite töödest selgub kliendi koolitamise olulisus tarkvaraprojekti edukuse tagamisel (Coombs *et al* 1999: 150). Koolitamine mõjutab kliendi suhtumist tarkvarasse. Koolitamise käigus omandab klient vajalikud oskused ja teadmised tarkvara kasutamiseks. See suurendab kliendi enesekindlust tarkvara kasutamise osas ning muudab tarkvara kliendile vastuvõetavamaks. Koolitamist saab kasutada ka veenmaks kliente tarkvara kasutama (Wilson *et al* 1999: 115). Üldiselt koolitatakse klienti peale tarkvara valmis arendamist, kuid on leitud, et klientide koolitamine tarkvara arendamise protsessi ajal paneb kliente rohkem tarkvara arendamisse panustama (Jiang *et al* 1998: 933).

Muudatuste juhtimine hõlmab endas tarkvara juurutamist. Tarkvara kliendile tutvustamine võib tuua kaasa olulisel hulgal muudatusi ja panna paljud kasutajad uude situatsiooni. Muudatustest tulenevalt võivad kasutajad tunda negatiivseid emotsioone. Potentsiaalne vastuhakk suureneb kui kasutajal on madal vastuvõtlikkus muutustele ja muutus mõjutab kliendi ettevõtet suurel määral (Butler 2003: 217). Tarkvara arendava ettevõtte juhatus peab projekti osavalt haldama, loomaks sobivad tingimused muutusteks kliendi poolel ning vältimaks negatiivsete emotsioonide tekkimist. (Butler *et al* 1999: 368).

Kokkuvõttes tuleb töö- ja palgaarvestustarkvara arendamise jätkusuutlikkuse tagamiseks viia läbi projekte klientidega ning säilitada konkurentsieelis. Töö- ja palgaarvestustarkvara pakkumine kliendile algab kliendi valmisoleku hindamisega ning sellele järgnevalt läbitakse süstemaatiliselt erinevad etapid projekti edukaks läbiviimiseks. Tähtsustatakse sealjuures realistlikku kulude hindamist, sest projektide puhul võivad kulud planeeritutest kordades erineda kui puudub korralik planeerimine (Xu *et al* 2014: 291). Lisaks on oluline pakkuda kliendile igakülgset tuge uute funktsionaalsuste lisamisel. Oluliselt mõjutab töö- ja palgaarvestustarkvara kliendile pakkumist töö- ja palgaarvestustarkvara arendava ettevõtte konkurentsieelis. Järgides hästi tuntud printsiipe ja tavasid on töö- ja palgaarvestustarkvara arendatavatel ettevõtetel võimalik luua stabiilne kliendibaas ning juurutada edukalt oma tarkvara ning uusi arendusi kliendipoolel.

2. TÖÖ- JA PALGAARVESTUSTARKVARADE ARENDAMISE ANALÜÜS

2.1. Uurimismetoodika ja tarkvarade tutvustus

Järgnevas peatükis kirjeldab autor valimis olevaid ettevõtteid, põhjendab uurimismetoodika valikut ning annab ülevaate töö- ja palgaarvestustarkvaradest.

Autor valis uurimismeetodiks juhtumiuuringu ja analüüsis kümne ettevõtte tarkvaraarenduse praktikat. Uurimismetoodika valikul tugines autor soovile katta võimalikult suur osa töö- ja palgaarvestustarkvara arendavate ettevõtete sektorist Eestis ning ühtlasi uurida iga kaetud ettevõtet põhjalikult. Juhtumisuuringus nõustus osalema 10 ettevõtet, mis moodustavad suurema osa Eesti töö- ja palgaarvestustarkvara pakkuvatest ettevõtetest.

Valimiks valis autor 12 tarkvaraettevõtet, kelle poolt pakutavate tarkvarade seas leidub töö- ja/või palgaarvestustarkvara. Valimisse valitud ettevõtetest 10 nõustusid intervjuus osalema. Intervjuus osalenud ettevõtete poolt pakutavate tarkvarade logodest annab ülevaate joonis 5. Tarkvaraturu suurematest tegijatest katavad vastanud hinnanguliselt 75% luues väga hea kaetuse. Ettevõtteid valimisse valides püüti tagada ettevõtete varieeruvus erinevate parameetrite põhjal. Vastanute seas on nii suuri kui väikeseid ettevõtteid, lühikese (4.a.) kui ka üle 24.a. tegutsemise ajalooga ettevõtted.

Joonis 5. Intervjuul osalenud tarkvarade logod (autori koostatud)

Andmete kogumise peamiseks meetodiks valis autor intervjuud. Lisaks kasutas autor ettevõtete veebilehtedel olevat informatsiooni. Valik tulenes vajadusest saada põhjalikud vastused intervjuuküsimustele. Antud juhul ostus sobivaks ettevõtete esindajatega kohtumine ning intervjuu läbiviimine. Selline lähenemine võimaldab intervjueeritaval iga küsimust kommenteerida vastavalt vajadusele ning intervjueerijal täpsustusi teha intervjuu käigus.

Intervjuu plaani koostades (vt lisa 1) lähtus autor bakalaureusetöö eesmärgist. Intervjuu loogika on anda ülevaade tarkvara arendava ettevõtte üldisest poliitikast, tarkvaraarenduse nüanssidest ning arendust mõjutavatest teguritest. Intervjuu küsimused sai jagatud eeltoodud loogika põhjal kolme plokki. Esimene plokk küsimusi annab ülevaate põhjustest kõnealuse tarkvara arendamiseks. Lisaks uuritakse tarkvara iseloomujoonte ning konkurentsieeliste kohta. Oluline on tarkvara tulevikuvisiooni

käsitlemine esimeses küsimuste plokis. See täpsustab ettevõtte poliitikat. Teine plokk küsimusi uurib tarkvaraarenduse nüansse. Täpsemalt uuritakse tarkvaraarenduse tööetappe, nende olulisust, kasutatavat arendusmetoodikat, dokumenteerimise protsessi, tiimitööd, arendusprobleeme ja uuenduste juurutamise protsessi. Viimane plokk küsimusi loob ülevaate kliendiga suhtlemisest nii tarkvaraarenduse protsessi raames kui ka juurutamise ning hooldamise raames. Viimase küsimuste ploki eesmärk on luua üldpilt suhtlusest kliendiga ning panustamisest kliendi murede kuulamisse ning kliendi koolitamisse/aitamisse.

Intervjuu puhul tuuakse välja järgmised puudused (Intervjuu … 2016):

- intervjuu võtab palju aega,
- eeldab hoolikat kavandamist ning intervjuu läbiviimise õppimist,
- intervjuu transkribeerimine ja analüüs võtab palju aega,
- intervjueeritava ja intervjueerija vastastikune suhe võib mõjutada tulemusi,
- usaldusväärsust võib nõrgendada intervjueeritava kalduvus anda sotsiaalselt soovitavaid vastuseid või luua endast muljet.

Suurimaks puuduseks antud juhul võib lugeda vastuste tõesuse. Raske on kontrollida intervjueeritava poolt antavate vastuste tõesust – intervjueeritava poolt jagatav informatsioon ei ole suures osas teistest allikatest leitav. Sellele tuginedes eeldatakse intervjueeritava poolt tõese informatsiooni jagamist intervjuu käigus.

Valimisse valitud ettevõtetega võeti ühendust ning avaldati soovi intervjuuks. Kümne ettevõtte esindajaga lepiti kokku koht ja aeg intervjuu läbiviimiseks. Tabel 1 annab ülevaate intervjuudeks kokkulepitud kuupäevadest ning intervjuude kestvustest. Tabelis on välja toodud iga intervjuu raames ettevõte, arendatav tarkvara ning ettevõtte esindaja. Intervjuule eelnevalt tutvus autor ettevõtte poolt pakutava tarkvara koduleheküljega luues üldise pildi ettevõtte kohta. Koduleheküljed sisaldasid sarnast informatsiooni tarkvarade lõikes – autor kogus koduleheküljelt informatsiooni tarkvara funktsionaalsuse, hinna, klienditoe olemasolu ja moodulite kohta. Lähtuti töö- ja/või palgaarvestuse mooduli ülesehitusest tarkvara üldistamisel. Sel viisil tagas autor kui intervjueerija teatud baasi olemasolu. Kodulehekülgedega tutvumine võimaldas veenduda vastuste tõesuses kontrollides intervjuu käigus jagatud informatsiooni vastavust kodulehel olevale

informatsioonile. Andmete põhikogumine toimus läbi intervjuude. Intervjuude transkriptsioonid on toodud välja lisades.

Tabel 1. Intervjuul osalenud ettevõtte tarkvara ja esindaja, intervjuu kuupäev ja kestvus

Tarkvara	**Ettevõtte nimi**	**Esindaja**	**Kuupäev**	**Kestvus**
Merit	Merit Tarkvara AS	Andres Kert	14.03.2016	2 tundi
Erply BOOKS	Margn OÜ	Taavi Hõbejõgi	21.03.2016	1 tund
NOOM	Astro Baltics OÜ	Erkki Ergma	22.03.2016	2 tundi
Persona	Fujitsu Estonia AS	Kaidi Neeme	28.03.2016	1 tund
SimplBooks	SimplBooks OÜ	Jaanus Reismaa	29.03.2016	1 tund
Toggl	Toggl OÜ	Alari Aho	29.03.2016	1 tund
RVSoft	RVSoft OÜ	Urve Överus	30.03.2016	1 tund
Tresoor	Tresoor Tarkvara OÜ	Kaili Saar	30.03.2016	1 tund
Begin	Begin OÜ	Siim Laurik	1.04.2016	2 tundi
Profit	Intellisoft OÜ	Jaana Tihhanovski	4.04.2016	1 tundi
			Kokku	**13 tundi**

Allikas: autori koostatud.

Tabel 2 tutvustab lühidalt valimis käsitletud ettevõtete vanust ning 2014. aasta seisuga müügitulu ja ärikasumit. Tabeli koostamisel on lähtutud ettevõtete majandusaasta aruannetes esitatud andmetest. Töö edasistes osades pole eelnimetatud dokumentatsiooni kasutatud. Antud töö keskendub sisemistele arenguprotsessidele ja sellest tulenevalt kasutatakse vaid intervjuu käigus omandatud informatsiooni edasiste analüüside läbi viimisel. Fujitsu Estonia AS põhitegevus ei ole tarkvara Persona arendamine ja sellest tulenevalt müügitulu ja ärikasumit tabelis välja ei toodud. Tabeli põhjal omab Toggl OÜ suurimat müügitulu ja ärikasumit. Margn OÜ müügitulu on kõige tagasihoidlikum ning SimplBooks OÜ ärikasum on ainsana negatiivne. Kõige vanem ettevõte vastanutest on Merit Tarkvara AS vanusega 24 aastat 5 kuud. Kõige noorem vastanud ettevõte on SimplBooks OÜ vanusega 4 aastat 2 kuud. Ettevõtete keskmine vanus on 13 aastat.

Tabel 2. Intervjuul osalenud ettevõte, asutamise kuupäev, 2014 aasta müügitulu ja ärikasum

Ettevõte	Asutamise kuupäev	Müügitulu (EUR)	Ärikasum (EUR)
Merit Tarkvara AS	30.10.1991	1 509 886	337 123
Margn OÜ	2.08.2011	27 185	7 293
Astro Baltics OÜ	11.03.1998	1 302 738	23 417
Fujitsu Estonia AS	-	-	-
SimplBooks OÜ	4.01.2012	78 388	-514
Toggl OÜ	7.02.2007	1 813 612	777 184
RVSoft OÜ	27.12.1990	253 705	83 866
Tresoor Tarkvara OÜ	26.10.1998	141 961	11 212
Begin OÜ	3.05.2011	141 799	13 182
Intellisoft OÜ	18.04.2002	116 302	39 765

Allikas: (Äripäeva Infopank … 2016).

Kokkuvõtvalt võib alapeatüki põhjal öelda, et valimisse kuulus 12 Eesti tarkvaraettevõtet. Valimisse valitud ettevõtetest 10 nõustusid intervjuus osalema. Andmete kogumise peamiseks meetodiks olid intervjuud. Lisaks kasutati ettevõtete veebilehtedel olevat informatsiooni. Kõik valimisse kuuluvad ettevõtted arendavad töö- ja/või palgaarvestustarkvara.

2.2. Tarkvaraarenduse andmete analüüs

Järgnevas alapeatükis analüüsitakse erinevate ettevõtete vastuseid intervjuuküsimustele ning tuuakse välja sarnasused ning märksõnad erinevate ettevõtete lähenemistes. Analüüsis kasutatakse veebilehtede analüüsi ning analüüsis välja toodud andmed pärinevad intervjuudest.

Intervjuul osalenud ettevõtete poolt pakutakse tarkvara, mis sisaldab töö- ja/või palgaarvestuse moodulit või funktsionaalsust. Järgnevalt analüüsitakse intervjuudest kogutud informatsiooni toomaks välja ning analüüsimaks sarnasusi tarkvarade lõikes.

Kõik uuritud tarkvarad said arendatud lähtuvalt turunõudluse või ettevõtte sisemise vajaduse olemasolust. Erandiks oli Fujitsu Estonia AS poolt arendatud palgaarvestust pakkuv tarkvara Persona, mille arendamise alustamise ajendiks oli riigihange. Ettevõte

sisemisest vajadusest arendatud tarkvarad olid algselt kasutamiseks arendaja enda vajaduste rahuldamiseks. Hiljem muutusid antud tarkvarad äriks. Ülevaate erinevate tarkvarade arendamise alustamise ajenditest annab tabel 3.

Tabel 3. Ajend tarkvara arendamise alustamiseks

Tarkvara	**Ajend**
Merit	turunõudlus
Erply BOOKS	turunõudlus
NOOM	ettevõtte sisemine vajadus
Persona	riigihange
SimplBooks	ettevõtte sisemine vajadus
Toggl	turunõudlus
RVSoft	turunõudlus
Tresoor	turunõudlus
Begin	turunõudlus
Profit	ettevõtte sisemine vajadus

Allikas: autori koostatud.

Kogutud andmete põhjal iseloomustavad ettevõtted oma tarkvara kasutust peamiselt märksõnadega lihtsus ja paindlikkus. Tegu on konfliktsete märksõnadega ja sellest tulenevalt keskendutakse peamiselt ühele neist märksõnadest tarkvara arendamisel. Erandiks on Erply BOOKS, Begin ja Profit, mis pakuvad lihtsat lahendust suurele osale klientidest, kuid võivad luua paindliku ja keerulise erilahenduse suuremale kliendile. Lisaks on tarkvarade puhul toodud välja märksõnad automaatsus, laiapõhjalisus, kaasaegsus, kasutajasõbralikkus, mobiilsus, võimsus, multifunktsionaalsus ja veebipõhisus. Ülevaate tarkvarade kasutust iseloomustavatest märksõnadest annab tabel 4.

Tabel 4. Tarkvara kasutust iseloomustavad märksõnad (X tähistab vastava märksõna kasutust)

Tarkvara	**Lihtsus**	**Paindlikkus**	**Muud märksõnad**
Merit	X		
Erply BOOKS	X	X	automaatsus
NOOM		X	laiapõhjalisus
Persona			kaasaegsus, kasutajasõbralikkus
SimplBooks	X		mobiilsus
Toggl	X		võimsus
RVSoft		X	
Tresoor		X	multifunktsionaalsus
Begin	X	X	veebipõhisus
Profit	X	X	

Allikas: autori koostatud.

Kõik käsitletud tarkvarad on tiheda konkurentsiga turul mõningate eranditega - NOOM ja Begin on oma valdkonnas konkurentsitus seisus. NOOM pakub ainsana täielikult ettevõtte äriprotsessidega kohanevat erilahendust ning omab monopoli tanklakettide poolt kasutatava tarkvara turul. Begin pakub tööarvestuse vallas terviklahendust ning on sellega konkurentsitus seisus. Iga tarkvara puhul on konkurentsieelis, mis teda konkurentidest eristab. Lühidalt on toodud välja iga tarkvara erisus konkurentidega võrreldes tabelis 5.

Tabel 5. Tarkvara konkurentidest eristavad tegurid

Tarkvara	**Konkurentidest eristav tegur**
Merit	keskendunud väikeste/keskmistele ettevõtetele, professionaalne klienditugi
Erply BOOKS	konkurentidest parem tehisintellekt
NOOM	paindlikkus täielikult kliendi protsessidele vastav lahendus luua
Persona	ainus personali/tööarvestuse/palgaarvestuse veebipõhine terviklahendus
SimplBooks	lihtsus (koolitust pole kasutamiseks vaja)
Toggl	tööarvestus üksikisikutele
RVSoft	võrguvaba klient-server lahendus turvalisuse säilitamiseks
Tresoor	pikaajaline kogemus
Begin	tööarvestuse erinevate viiside terviklahendus
Profit	väljaostu võimalusega (ei ole igakuist/iga-aastast makset)

Allikas: autori koostatud.

Iga tarkvara puhul toodi välja moodul, mille arendamiseks palju ressursse kulunud. Meriti puhul on selleks klienditugi, Erply BOOKSi puhul pangaimport, NOOMi puhul laondus, Persona puhul puhkusearvestuse moodul, SimplBooksil palgaarvestus, Tresooril raamatupidamine, Beginil tööaja registreerimine, Profitil ka raamatupidamine. Toggl ja RVSoft ei osanud ühte moodulit välja tuua, sest kõigele on palju ressursse kulunud. Uuritud tarkvaradel on tulevikuvisioonid - kuhu tulevikus ressursid suunatakse ja mida arendatakse. Keskseteks märksõnadeks osutus uute moodulite arendamine, terve platvormi uuendamine, integratsioonide juurde loomine ning tarkvara pilvepõhiseks muutmine. Erply BOOKS ja NOOM soovivad olla andmevoolude keskpunkt, mis väljendub soovis pakkuda ühtset terviklahendust kõigi kliendi äriprotsesside katmiseks kaotades ära vajaduse erinevate tarkvarade järele. Tarkvarade tulevikueesmärkidest annab ülevaate tabel 6.

Tabel 6. Tarkvara eesmärgid tulevikuks

Tarkvara	**Pilv**	**Integratsioonid**	**Uued moodulid**	**Platvormi uuendamine**	**Andmevoolude keskpunkt**
Merit	X	X			
Erply BOOKS		X			X
NOOM	X	X			X
Persona			X		
SimplBooks			X		
Toggl			X		
RVSoft				X	
Tresoor			X		
Begin				X	
Profit				X	

Allikas: autori koostatud.

Tarkvaraarendusele toetudes uuriti ettevõtetelt nende poolt praktiseeritavate tarkvaraarenduse tööetappide kohta. Järgnevalt tuuakse ettevõtete lõikes välja kasutuses tarkvaraarenduse tööetapid ning tiimitöö, dokumentatsiooni ja tarkvaraarenduse meetodite eripärad.

Meriti arendamisel praktiseeritakse aastatega välja kujunenud metodoloogiat ning läbitakse järgnevad tööetapid:

1) arendusjuhi poolt kogutakse kokku ideed arenduseks,

2) arutatakse kogutud ideede tehnilist teostatavust,
    a. koosolekul räägitakse lahti detailid,
    b. leitakse kõige paremad lahendused,
    c. pannakse kirja lahendused,
3) pannakse ideede lahendused prioriteetidega järjekorda,
4) võetakse nimekirjast prioriteetide järjekorras ideed ja arendatakse lahendus.

Kõige olulisem ja enim ressursse nõudev on etapp 1. ehk ideede kogumine ja sõelumine järgmiseks etapiks. Kindel arendusmetoodika puudub, kuid kirjeldus sarnaneb XP variatsiooniga. Koosolekuid tehakse harva. Vahel eraldatakse tund arutamaks ülesannete täitmist. Kõike arutatakse jooksvalt ning kõigil arendajatel on õigus kaasa rääkida. Ülesande püstitaja testib ülesande lahendust. Tarkvaramuudatused dokumenteeritakse ja organiseerimiseks kasutatakse Githubi ja FogBugzi keskkondi. Tarkvara uuendatakse iganädalaselt ja juurutatakse kolmes etapis, mis uue tarkvara versiooni puhul läbitakse: supertest (elav arendus), test (mõnede klientide poolt kasutatav), live (mõjutab kõiki kliente).

Erply BOOKSi arendamisel kasutatakse paralleelarenduse metodoloogiat ning läbitakse järgnevad tööetapid:

1) planeerimine,
2) analüüs,
3) prioriteetide järjekorras disain ja arendamine,
    a. esiteks arendatakse lahendus tähtsatele vigadele,
    b. teiseks arendatakse tähtsad tarkvarauuendused,
    c. viimaks arendatakse lahend vähetähtsatele vigadele.

Kõige olulisem ja enim ressursse nõudev on etapp 1.a. ehk tarkvarauuenduste arendamine. Paralleelselt käib töö etappides 1.a ja 1.b. piisavate ressursside olemasolul. Arendusmetoodikana kasutatakse paarisprogrammeerimist – tegu XP ühe osaga. Tarkvaramuudatused dokumenteeritakse ja organiseerituse hoidmiseks kasutatakse Bitbucketi keskkonda. Pilvepõhist tarkvara uuendatakse iganädalaselt ja tarkvara juurutatakse pannes rõhku inimeste harjumuste muutmisele. Juurutatakse pannes uuenduse peale algul mõnele kliendile ja hiljem rohkematele.

NOOMi arendamisel tehakse projekt ja kasutatakse projektijuhi ressurssi. Klientidele lähenetakse individuaalselt ning läbitakse järgnevad tööetapid:

1) püstitatakse probleem projektijuhi ja kliendi esindaja kokkuleppel;
2) koos käiakse läbi ettevõtte tööprotsessid ja tehakse mudelid;
3) kaardistuse tulemusel luuakse tööprotsessi dokument;
4) pannakse paika kliendi poolt vajatavad moodulid ning nende arendamiseks ja juurutamiseks vajalikud etapid koos ajaliiniga projekti teostamiseks;
5) arendatakse projektipõhiselt kliendile lahendus;
6) kliendile antakse lahendus üle ja hakatakse juurutama.

Kõige olulisemad ja enim ressursse nõudvad on etapid 2. and 3. ehk tööprotsesside mudelite koostamine ja kaardistus. Arendusmetoodikana kasutatakse jooksvat standardit ehk XP. Paralleelselt arendatakse erilahendust, standardlahendust ja tegeletakse hooldusega. Tarkvaramuudatused dokumenteeritakse versiooniuuenduste dokumenti, mis saadetakse ka kliendile. Tarkvara uuendatakse vastavalt kliendi poolt nõutavatele funktsioonidele ja igale kliendile juurutatakse tarkvara eraldi. Juurutatakse esimesel korral ligikaudu 6 kuud. Uudse lähenemisega tagatakse igale kliendile igakuine maht ettevõtte ressursse NOOMi lahenduse edasiseks arendamiseks.

Persona arendamisel tehakse tarkvaraprojekt ning läbitakse järgnevad tööetapid:

1) algatusfaas,
2) loomisfaas,
3) arendusfaas,
4) juurutusfaas.

Kõik tööetapid on ühtlaselt olulised ja ressursse nõudvad. Arendusmetoodikana kasutatakse Fujitsu enda kujundatud tarkvaraarendusmetoodikat, mis on iteratiivne ja järkjärguline ning sarnane SCRUMile. Tarkvaramuudatused dokumenteeritakse ning tarkvarauuendusi tehakse võimalikult sageli. See vähendab uuendusest tulenevate probleemide tõenäosust ja sunnib protsesse järgima.

SimplBooksi arendamisel läbitakse järgnevad tööetapid:

1) analüüsitakse klientide tagasisidet ja pannakse paika vajadused,
2) analüüsitakse vajadusi,
3) arendatakse lahendus vajadusele,
4) testitakse lahendust,
5) antakse lahendus kliendile üle.

Kõige olulisem ja ressursimahukam etapp on 2. ehk analüüs. Analüüsi etapid tagatakse, et kõigi teguritega arvestatakse ning tehakse lahendus esimese korraga võimalikult hästi ära. Arendusmetoodikana kasutatakse paarisprogrammeerimist ehk XP osa. Tarkvaramuudatuste dokumentatsiooni ei koostada, kuid organiseerituseks kasutatakse Githubi keskkonda. Struktuurist tulenevalt kulutatakse paar tundi kõigile klientidele uue tarkvara versiooni peale panemiseks ning uuendusi tehakse igakuiselt. Juurutamist ei eksisteeri ja tarkvara on kasutatav ilma koolituse/juhendita.

Toggli arendamisel on kujunenud välja oma printsiibid ja etapid. Läbitakse järgnevad tööetapid:

1) planeerimine,
2) analüüs ja tagasiside,
3) testimine,
4) juurutamine,
5) hooldus.

Kõik etapid on olulised ja ressursimahukad eesmärgi saavutamise kontekstis. Arendusmetoodika on agiilne ning täpsemalt SCRUM. Tarkvaramuudatused dokumenteeritakse avalikult koduleheküljele ja organiseerituse hoidmiseks kasutatakse Githubi keskkonda. Uuendusi tehakse pidevalt ja püütakse uuendus kasutajani saada võimalikult kiiresti. Juurutamist ei ole tarkvara lihtsusest tulenevalt.

RVSoft arendamisel läbitakse järgnevad tööetapid:

1) selgitatakse välja kliendi vajadused,
2) projekteerija poolt luuakse tarkvaraprojekt,
3) arendatakse lahendus,
4) testitakse lahendust,

5) antakse lahendus kliendile üle,
6) muudetakse lahendus vastavalt kliendi vajadusele.

Kõige olulisem ja ressursimahukam on kliendiga suhtlemine ja tagasiside saamine, sest tagasiside peab olema mõistlik, selge ning ei tohi tugineda emotsioonidel. Arendusmetoodikana kasutatakse SCRUMi - iga nädal toimub koosolek, kus pannakse plaan paika arendamiseks ning viiakse tiim kurssi. Dokumenteeritakse kõik tarkvaramuudatused, kuid ühtset platvormi organiseerituse tagamiseks ei ole. Igale kliendile lähenetakse individuaalselt ja sellest tulenevalt tarkvara uuendamise aeg ja juurutamine sõltub igast kliendist eraldi.

Tresoori arendamisel läbitakse järgnevad tööetapid:

1) analüüs,
2) disain,
3) arendamine,
4) juurutamine.

Kõige olulisem ja ressursimahukam tööetapp on arendamine. Arendusmetoodikana kasutatakse XP-d. Suuremad tööd püütakse arendada valmis ühe korraga, kuid kõrvale tehakse ka väiksemaid töid. Tihti leitakse tarkvarast vead, mis lahendatakse kiiresti vajadusest tulenevalt. Dokumenteeritakse kõik tarkvaramuudatused, kuid ühtset platvormi organiseerituse tagamiseks ei ole. Dokumentatsioon on tagaplaanil. Eelkõige tuleb lahendus kiirelt kliendini toimetada. Uuendamiseks peab klient ise tarkvara alla laadima ja installeerima ning klientidele pakutakse regulaarselt koolitusi.

Begini arendamisel läbitakse järgnevad tööetapid:

1) selgitatakse ühe kliendi vajadused,
2) arutatakse kas see aitaks ka teisi kliente,
3) planeeritakse sprint,
4) disainitakse,
5) arendatakse,
6) testitakse,
7) juurutatakse.

Kõige olulisem ja ressursimahukam on arendamine. Arendusmetoodikana kasutatakse SCRUMi. Dokumenteeritakse kõik tarkvaramuudatused ja organiseerituseks kasutatakse JIRA keskkonda. Tarkvaraga kaasneb 90 minutit koolitust juurutamiseks ning uuendusi tehakse iganädalaselt.

Profiti arendamisel läbitakse järgnevad tööetapid:

1) selgitatakse kliendi vajadused,
2) analüüsitakse vajadusi,
3) arendatakse,
4) testitakse,
5) juurutatakse.

Kõige olulisem ja ressursimahukam on vajaduste selgitamine ja analüüs. Arendusmetoodikana kasutatakse XP-d. Dokumenteeritakse kõik tarkvaramuudatused. Tarkvaraga kaasneb 2 tundi koolitust juurutamiseks ning uuendusi tehakse vastavalt kliendi vajadusele.

Iga tarkvara puhul on arendamisel läbitavad tööetapid erinevad, kuid kõigi vaadeldud tarkvarade puhul läbitakse kindlasti järgnevad tööetapid:

1) analüüs (nõuete väljaselgitamine),
2) arendamine (teostamine),
3) juurutamine (tugi).

Nelja tarkvara puhul on kõige olulisem ja ressursimahukam analüüsi etapp ehk kliendi vajaduste selgitamine ning analüüsimine. Samuti on nelja tarkvara puhul kõige olulisem ja ressursimahukam etapp arendamine, mis nõuab ressursse ja tähelepanu arendajatelt. Kaks tarkvara, Toggl ja Persona, tähtsustavad kõiki etappe ühtviisi. Kõik tarkvarad kasutavad arendamiseks agiilset metoodikat. Kuigi ühegi tarkvara puhul ei kasutata puhast versiooni XP-st või SCRUMist, siis on loodud oma tarkvara arendamiseks sobiv variatsioon ühest või teisest agiilsest arendusmetoodikast. Variatsiooni XP-st kasutab suurem osa tarkvarasid ehk kuus ning SCRUMi variatsiooni on juurutanud neli tarkvara. Tabel 7 tutvustab eelnevat tarkvarade lõikes.

Tabel 7. Tarkvara arendamise olulisim tööetapp ja arendusmetoodika

Tarkvara	Olulisim tööetapp	Arendusmetoodika (variatsioon)
Merit	analüüs	XP
Erply BOOKS	arendamine	XP
NOOM	analüüs	XP
Persona	kõik	SCRUM
SimplBooks	analüüs	XP
Toggl	kõik	SCRUM
RVSoft	analüüs	SCRUM
Tresoor	arendamine	XP
Begin	arendamine	SCRUM
Profit	analüüs	XP

Allikas: autori koostatud.

Tabel 8 põhjal kasutatakse tarkvaramuudatuste dokumenteerimiseks keskkondi Github, Bitbucket ja JIRA. Tarkvarauuendusi tehakse iganädalaselt või SimplBooksi puhul igakuiselt. Erandiks on tarkvarad, kus on erilahendused või uuendamine sõltub kliendist, sest tarkvara pole veebipõhine. Eelnimetatud tarkvarade puhul sõltub uuendamise sagedus suuresti kliendi vajadustest. Eelneva kohta on täpsem ülevaade välja toodud tabelis 8. Tiimitööd iseloomustab kõigis tarkvara tootvatest ettevõtetes jooksev standard ehk kõik räägivad kaasa ning suheldakse jooksvalt. Koosolekuid välditakse. Erandid on mõned tarkvarad nagu Begin ja NOOM, kus on ettevõttes kindlad protseduurid tiimitöö korraldamiseks. Igale töötajale ja tiimile määratakse kindlad ülesanded ning hoitakse sel viisil korda.

Tabel 8. Tarkvara arendamise dokumentatsioon ja uuendamise sagedus

Tarkvara	Dokumentatsioon (tarkvara)	Uuendamine
Merit	Github, FogBugz	iga nädal
Erply BOOKS	Bitbucket	iga nädal
NOOM	-	vastavalt kliendile
Persona	JIRA	iga nädal
SimplBooks	Github	igakuine
Toggl	Github	iga nädal
RVSoft	-	vastavalt kliendile
Tresoor	-	vastavalt kliendile
Begin	JIRA	iga nädal
Profit	-	vastavalt kliendile

Allikas: autori koostatud.

Tabel 9 annab ülevaate tarkvarade arendamisel tehtud vigadest ning eraldi käsitletakse nimetatud vigade vältimist tulevikus. Läbivaks temaatikaks on analüüsi faasis kliendist mööda rääkimine, millest tulenevalt hiljem valmis arendatud lahendus ei vasta kliendi algsetele soovidele. Antud vea vältimiseks tuleb arendamisele eelnevalt koostada kirjalik dokumentatsioon kliendi soovidest. Eelnevaga tagatakse paigas tegevuskava ning arenduse fookus. Arendamisel esinevaks veaks loetakse ebaefektiivsed protsessid, mis pärsivad efektiivset arendamist ning millest ülesaamiseks tuleb pidevalt protsesse parendada. Erply BOOKSi puhul luuakse enim vigu riiklike uuenduste raames. Riiklikest uuendustest teavitatakse lühikese etteteatamisega. Meriti puhul tehti suurim viga uuele platvormile liikudes. Tarkvarast loodi mitu varianti, millest käiku läks üks. Suur osa ressursse läks raisku.

Tabel 9. Tarkvara arendamisel tehtavad vead

Tarkvara	**Arendamise vead**
Merit	uuele platvormile üleminek
Erply BOOKS	riiklikud uuendused lühikese etteteatamisajaga
NOOM	dokumentatsiooni puudumine
Persona	ebaefektiivsed protsessid
SimplBooks	-
Toggl	arendus läheb üle tähtaja
RVSoft	möödarääkimine kliendist kirjaliku dokumentatsiooni puudumise tõttu
Tresoor	möödarääkimine kliendist kirjaliku dokumentatsiooni puudumise tõttu
Begin	möödarääkimine kliendist kirjaliku dokumentatsiooni puudumise tõttu
Profit	möödarääkimine kliendist kirjaliku dokumentatsiooni puudumise tõttu

Allikas: autori koostatud.

Tabel 10 annab ülevaate tarkvara arendavate ettevõtete suhtumisest klientidesse ning hinnastamise poliitikasse. Iga tarkvara puhul tuuakse lisaks välja kliendi rahulolu säilitamiseks tehtav pingutus. Klientide arendusse kaasamises jagunetakse kahte gruppi – individuaalne ja grupiviisiline. Grupiviisiliselt lähenevad tarkvarad, mille puhul tehakse tarkvarasse muudatus grupi klientide ühisel soovil. Individuaalselt lähenevad tarkvarad, mille puhul tehakse klientidele erilahendusi ehk klient saab täpsustada oma vajadused ning vastavalt kliendi vajadustele arendatakse lahendus. Valdav enamus tarkvarasid pakuvad koolitusi. Enamus tarkvarad pakuvad kliendile koolitust juurutamise käigus vähendamaks kliendi hilisemate küsimuste arvu ning tagamaks tarkvara kasutusmallidest

ülevaate. Kliendi rahulolu säilitamiseks püütakse pakkuda head kliendituge ning kuulata klientide soove. Lõpetuseks uuriti tarkvara hinnastamist. Kõige populaarsemad hinnastamise viisid on paketipõhine ning moodulitest või kasutajate arvust sõltuv hinnastamine. Lisaks on erilahendusi välja töötava tarkvara puhul iga projekt erinev ning sellest tulenevalt on iga projekti puhul unikaalne leping, millega määratakse hind.

Tabel 10. Tarkvara arendamise mõjutatus klientidest

Tarkvara	**Klientide arendusse kaasamine**	**Koolitused**	**Kliendi rahulolu säilitamine**	**Hinnastamine**
Merit	grupiviisiline	jah	eesliinil ja stabiilne	progresseeruv
Erply BOOKS	individuaalne/ grupiviisiline	jah	kaasaegne ja kiire tugi	moodulitest sõltuv
NOOM	individuaalne	jah	partnerleping kiireks toeks	lepingupõhine
Persona	grupiviisiline	jah	kiire tugi	suurusest ja moodulitest sõltuv
SimplBooks	grupiviisiline	ei	kiire tugi	üks hind
Toggl	grupiviisiline	ei	kuulame klienti	paketipõhine
RVSoft	individuaalne	jah	erilahenduste võimalus	kasutajate arv
Tresoor	individuaalne	jah	erilahenduste võimalus	lepingupõhine
Begin	grupiviisiline	jah	kuulame klienti	kasutajate arv
Profit	individuaalne/ grupiviisiline	jah	kiire tugi	paketipõhine

Allikas: autori koostatud.

Alapeatükis selgus, et erinevate töö- ja palgaarvestustarkvara arendavate ettevõtete praktikates leidub sarnasusi. Tarkvaraarendusel leidub sarnasusi tarkvaraarendusmeetodite kasutamisel ning arendamise tööetappides. Klientide arendusse kaasamisel leidub samuti sarnasusi ettevõtete lõikes. Järgmises peatükis määratletakse enim kasutatud praktika leitud sarnasustele tuginedes.

2.3. Enim kasutatud praktika määratlus

Kogutud ja analüüsitud andmete põhjal saab määratleda enim kasutatud praktika. Käesolevas peatükis määratleb autor analüüsile tuginedes enim kasutatud praktika töö- ja palgaarvestustarkvara arendamiseks.

Enim kasutatud praktika töö- ja palgaarvestusetarkvara arendamise alustamiseks on esiteks veenduda turunõudluse olemasolus. Alternatiivne võimalus on veenduda ettevõtte sisemises vajaduses. Järgmiseks tuleb analüüsida olemasolevaid tarkvarasid ning luua üldpilt konkurentsist. Arendatav tarkvara peab erinema konkurentidest – intervjuude tulemusena selgus konkurentsieelise olemasolu tähtsus.

Töö- ja palgaarvestustarkvara arendamisele eelnevalt tuleb leida kliente ning selgitada ja analüüsida klientide vajadusi. Oluline on kuulata kliente ja lähtuda tarkvara arendamisel klientide vajadustest. Selgelt tuleb dokumenteerida klientide vajadused. Analüüsist selgus, et suurim viga on arendada tarkvara ilma kliente kaasamata – algusfaasis ei tohi otsustada kliendi eest, mida klient tahab.

Enamus vastajatest rõhutas, et tarkvara arendamisel on tähtsad tarkvara lihtsus ja paindlikkus. Enim kasutatud praktikast lähtuvalt panustada kliendi jaoks lihtsa tarkvara arendamisele või kõigi klientide vajadustele vastava paindliku tarkvara arendamisele. Arendamise käigus tuleb keskenduda kliendile tarkvara kiiresti kätte toimetamisele. Testimine on tagaplaanil. Enim kasutatud praktikast lähtuvalt keskenduda arendamisele ja juurutamisele ning käsitleda testimise tööetappi teisejärgulisena. Arendusmetoodikana tuleb kasutada agiilset tarkvaraarendusmeetodit – XP on enim levinud praktika ning võimaldab kohanduda jooksvalt tarkvaras tekkivate vigade parandamisele uute arenduste kõrvalt. Arendamisel on oluline tiimitöö – jooksvalt tuleb hoida kogu tiim kursis eesmärkidega ning arenduses olevate töödega. Kõik muudatused tuleb dokumenteerida kasutades veebikeskkonda ning tarkvara tuleb uuendada iganädalaselt.

Juurutamise faasis tuleb klienti koolitada selgitamaks kliendile tarkvara kasutusmalle ning tagamaks vastused hiljem tekkivatele küsimustele. Kliendi suurusest sõltuvalt tuleb soovitada kliendile koolitust - mida suurem klient, seda rohkem kasu koolitusest. Arendaja säästab ressursse kompetentsete klientide arvelt. Hiljem tuleb panustada

professionaalsesse klienditoesse ning kliendi soovide arvesse võtmisse säilitamaks kliente pikemas perspektiivis.

Kui klientide poolt on tarkvara juurutatud, siis muutub suhtumine arenduseesmärkide püstitamisse. Klientide kuulamise olulisus väheneb ning töö- ja palgaarvestustarkvara arendaja peab ise teadma, mida klient vajab. Kui just grupp kliente ei soovi uut arendust, siis otsustatakse tehtavad arendused kliendi eest. Klient ei oska tihti kriitilisi uuendusi soovida – näiteks riiklikud uuendused.

Suuremate klientide jaoks arendatakse erilahendusi. Enim kasutatud praktika on sellisel juhul kindlasti dokumenteerida kõik kliendi soovid ja vormistada kirjalikult projekt ning vastavad lepingud. See väldib hilisemaid vaidlusi kliendi ja tarkvara arendaja vahel. Levinud praktika on teha klientidega teeninduslepingud määramaks igale kliendile prioriteedi.

Viimaks tuleb hinnastamises selgus leida – enim kasutatud praktika on kliendile tarkvara hinna määramine töö- ja palgaarvestustarkvara kasutajate arvu ja moodulite arvu põhjal. Oluline on klientidele pidev arve esitamine ehk klientidelt tuleb võtta kuutasu. Enne stabiilse kliendibaasi saavutamist tuleb teha agressiivset müügitööd ning võimaluse korral tarkvara litsents maha müüa enne kui tarkvara ise eksisteerib.

Kokkuvõttes tuleb enim kasutatud praktikast lähtuvalt töö- ja palgaarvestustarkvara arendamisel kasutada agiilset tarkvaraarendusmeetodit, soovitatavalt ekstreemprogrammeerimist. Arendamisel tuleb läbida analüüsi, arendamise ja juurutamise tööetapid. Oluline on läbida põhjalikult analüüsi tööetapp. Kliente tuleb kuulata ja arendusse kaasata. Kõik soovid, lepped ja arendused peab dokumenteerima. Kliente tuleb koolitada ning eelistatud on kuutasul põhinev hinnastamine.

2.4. Wemply tarkvara edasiarendamise võimalused

Wemply on tarkvara töö- ja palgaarvestuse automatiseerimiseks, mis arendati eesmärgiga luua tarkvara spetsiifilise puidutööstusettevõtte tööarvestuse vajaduste lahendamiseks. Wemply arendamise ajendiks oli puidutööstusettevõtte vajadus spetsiifilise

tööarvestustarkvara järgi. Töö autor arendas antud vajadustele ja nõuetele vastava tööarvestustarkvara. Tarkvara iseloomustavateks märksõnadeks said lihtsus ja efektiivsus. Järgnevalt vaadeldakse iga teoorias välja toodud tarkvaraarenduse tööetappi Wemply kontekstis.

Nõuete väljaselgitamise etapis dokumenteeriti järgmised tarkvarale esitatavad nõuded:

- tööle asumise algus- ja lõpuaja logimine,
- töötaja asukoha logimine,
- lihtne palgaarvestus,
- erinevate raportite kasutamise/lisamise võimalus,
- kasutaja õiguste süsteem turvalisuse tagamiseks,
- veebipõhine.

Kavandamise etapis loodi tarkvarast prototüüp täitmaks lihtsamaid tarkvarale esitatud nõudeid. Välja selgitati prototüübi puudused ning alustati arendamist.

Tarkvara arendati SCRUM arendusmeetodile tuginedes. Koostati nimekiri arendatavatest funktsionaalsustest, arendati valmis funktsionaalsused ning koostati järgmine nimekiri. Arendamisega alustati 2.11.2014.

Testimise etapis koostati erinevaid teste eesmärgiga tuvastada vigu tarkvaras. Kaeti kõik tarkvara kasutusjuhud ning peale suuremaid muudatusi tarkvaras jooksutati koostatud teste. Leitud vigade korral vead parandati.

Juurutamise faasi Wemply edukalt ei ole läbinud ning turundamisele pole rõhku pandud. Wemply püüab erineda teistest turul olevatest konkurentidest oma unikaalse moodusega tööaegade registreerimiseks koos asukohaga. See võimaldab tööandjatel omada ülevaadet töötajate objektil veedetud tööaegadest ning samas veenduda, et tööaega salvestades oli töötaja objektil.

Edasi vaadeldakse Wemply arendamist enim kasutatud praktikale tuginedes. Sellele tuginedes leitakse üldised enim kasutatud põhimõtted Wemply edasiarendmaiseks.

Wemply arendamise ajendiks oli autori huvi ja tööarvestustarkvara vajav ettevõte. Turunõudlust ja turupilti arendamisele eelnevalt ei uuritud. Enim kasutatud praktika järgi oleks pidanud veenduma turunõudluses, mida antud juhul ei tehtud.

Turul esineva konkurentsiga tutvuti. Selgitati välja Wemply't konkurentidest eristav funktsionaalsus ehk konkurentsieelis. See läheb kokku enim kasutatud praktikaga.

Uue tarkvara arendamine algab klientide kuulamisest – enim kasutatud praktikast tulenevalt tuleks määrata kliendi soovide põhjal tarkvara disain ja funktsionaalsuses. Wemply puhul klienti ei kuulatud ning tarkvara arendati lähtuvalt arendaja arvamusest. Arendaja arvas kliendi eest mida klient vajab. Klienti ei kaastatud tarkvara arendamisse ning selle tulemusena kliendile lõpptulemus ei meeldinud.

Wemply arendamisel rõhuti tarkvara lihtsusele, kuid samas püüti võimalikult palju funktsionaalsusi panna kompaktsesse komplekti. Enim kasutatud praktika põhjal töö- ja palgaarvestustarkvarad peaksid rõhuma lihtsusele või paindlikkusele.

Arendamisel keskenduti Wemply puhul tarkvarale funktsionaalsuste lisamisele ning pidevale testimisele. Tarkvara turule pakkumisega ei kiirustatud ning arendati kaks aastat tarkvarale uusi funktsioone, kuid müügi või turule pakkumisega ei tegeletud. Enim kasutatud praktika eeldab vastupidist – minimaalne toode tuleks valmis arendada ning turule suunata.

Wemply arendamise käigus püüti arendusmetoodikana juurutada agiilset variatsiooni SCRUMist ehk pandi kirja ülesanded ning järgnevad kaks nädalat tegeleti uute arendustega kirja pandust lähtuvalt. SCRUM oli ka enim kasutatud praktika määratlemisel välja toodud, kuid vähem populaarne kui XP. SCRUM teeb väikesel tiimil keerulisemaks jooksva vigade parandamise. Dokumenteerimiseks kasutati Githubi keskkonda, mida mainiti enim kasutatud praktikas.

Wemply't juurutamise faasis kliendile ei suudetud maha müüa, sest ei lähtutud enim kasutatud praktikast. Klienti ei koolitatud ning esmapilgul uue tarkvaraga tutvudes klient ei saanud aru tarkvara kasutusjuhtudest. Sellest tulenevalt kadus kliendipoolne huvi. Lisaks kliendi vajadusi ei dokumenteeritud. Sellest tulenevalt polnud nimekirja funktsionaalsustest, mis Wemply's oleksid pidanud olema kliendi perspektiivist.

Enim kasutatud praktikast erinevalt kavandati Wemply hinnastamine aastase tasu põhiseks, kuid kliendi kasutajate arvust sõltuvaks, mis on enim kasutatud praktikaga kooskõlas. Puudus müügitöö ja turundus ning ei leitud Wemply jaoks kliente.

Tulevikuvisioon on kooskõlas enim kasutatud praktikaga. Arendada Wemply veebipõhist tarkvara ja luua juurde integratsioone. Wemply muudetakse mooduliks olemasolevatele tarkvaradele.

Enim kasutatud praktikast lähtuvalt püütakse Wemply puhul teha tulevikus agressiivset müügitööd (helistatakse potentsiaalselt tarkvarast huvitatud ettevõtete juhtidele ning pakutakse, et tullakse Wemply tarkvara tutvustama) ning kuulata uute arenduste tegemisel eelkõige klienti. Lisaks panustatakse koolitusmaterjali koostamisse.

Kokkuvõttes on suurimad Wemply arendamisel tehtud vead järgmised:

- turunõudluses ei veendutud;
- kliente ei otsitud ja klientide vajadusi ei võetud kuulda;
- otsustati ise, mida klient tahab;
- keskenduti uute funktsionaalsuste arendamisele ja testimisele;
- analüüsi ja juurutamise faasid jäeti enim kasutatud praktika tööetappidest vahele;
- keskenduti pilveteenuse arendamisele ja ei loodud liideseid olemasolevate tarkvaradega suhtlemiseks.

Wemply näol arendati tiheda konkurentsiga turule uus tarkvara kuigi oleks võinud luua olemasolevatele tarkvaradele lisamooduli. Enim kasutatud praktika määratlemisel oli oluline märksõna, et ei tasu leiutada jalgratast kui jalgratas on juba olemas. Antud juhul püüti leiutada uuesti jalgratast ning sellega turule tulla. Vigadest on Wemply puhul õpitud ja keskendutakse ainult tööarvestuse arendamisele. Palgaarvestust enam ei arendata. Lisaks keskendutakse integratsioonide arendamisele, et Wemply oleks võimalik siduda olemasolevate tarkvaradega. Sel viisil ei sisene Wemply turule konkurendina, vaid lisamoodulina, mida olemasolevatele töö- ja palgaarvestustarkvaradele saab külge monteerida ning sel viisil tuua rohkem lisandväärtust lõpptarbijale.

KOKKUVÕTE

Efektiivne töö- ja palgaarvestus on oluline hoidmaks ettevõtte kulud madalad ning tulud kõrged. Paljud ettevõtted kasutavad töö- ja palgaarvestustarkvara efektiivsuse tõstmiseks. Sellest tulenevalt peab tarkvarasid pidevalt arendama. Edasisel arendamisel aitab tarkvarade lõikes enim kasutatud arendamise praktika tundmine.

Arvestuse infosüsteemid töötlevad andmeid informatsiooni saamiseks ettevõtte sise- ja välistarbijate jaoks. Informatsiooni saab kasutada aruandluses ja otsuste langetamisel. Arvestuse infosüsteemide automatiseerimine lihtsustab rutiinseid protsesse ning muudab arvestuse efektiivsemaks. Automatiseerimine hõlmab endas vastava tarkvara arendamist asendamaks manuaalse töö algoritmidega, mida jooksutatakse automaatselt tarkvara sees.

Tarkvaraarendusmeetoditena eristatakse traditsioonilisi ja agiilseid meetodeid. Agiilsetest meetoditest kirjeldatakse SCRUMi, ekstreemprogrammeerimist ja Crystalit. Tööetappide loetlemisel lähtutakse erinevate autorite poolt kokkulepitud nimekirjal, mis määratleb tarkvara arendamisel läbitavate tööetappidena nõuete väljaselgitamise, kavandamise, teostamise, testimise ja toe pakkumise.

Tarkvara arendavad ettevõtted puutuvad kokku erinevate teguritega, mis mõjutavad tarkvara arendamise protsessi. Tarkvara arendamise motivaatorina tuuakse välja konkurentsieelise mõiste. Tarkvara arendamist takistavate teguritena käsitletakse klientide kaasamise ja juurutamisega seotud probleeme. Tarkvara arendamist kiirendava tegurina käsitletakse korrektse ja läbimõeldud tarkvaraprojekti koostamist ja juhtimist. Erinevate autorite käsitlustest tuuakse kokkuvõtvalt välja viis enim arendamise protsessi mõjutavat tegurit: nõuete väljaselgitamine, projektijuhtimine, tarkvaraarendusmeetodi kasutus, kliendi kaasamine, kliendi koolitamine ja muudatuste juhtimine.

Bakalaureusetöö empiirilises osas analüüsiti töö uurimiseesmärgi täitmiseks erinevatelt töö- ja palgaarvestustarkvara arendavatelt ettevõtetelt intervjuude käigus kogutud andmeid. Analüüsi tulemusena määratleti enim kasutatud praktika töö- ja palgaarvestustarkvara arendamiseks. Enim kasutatud praktika määratlemisele järgnevalt tutvustati Wemply töö- ja palgaarvestustarkvara. Wemply on töö autori poolt arendatud tarkvara ja enim kasutatud praktikat kasutatakse Wemply edasiarendamiseks.

Valimis oli vaatluse all 12 töö- ja palgaarvestustarkvara arendavat Eesti ettevõtet. Neist 10 ettevõtet nõustusid antud töö teema raames intervjuud andma. Ettevõtted varieeruvad kliendibaasi, vanuse, töötajate arvu ja müügitulu poolest.

Intervjuudes kogutud andmete põhjal toodi välja sarnasused töö- ja palgaarvestustarkvara arendavate ettevõtete lähenemises arendamisele. Peamiselt ajendas ettevõtteid tarkvara arendama turunõudluse olemasolu. Tarkvarasid iseloomustatakse märksõnadega lihtsus või paindlikkus. Iga tarkvara arendav ettevõte omab konkurentsieelist. Üldiselt on töö- ja palgaarvestustarkvarade turul tihe konkurents. Tarkvarade arendamise eesmärgid tulevikuks on veebipõhise lahenduse parendamine, integratsioonide juurde arendamine, uute moodulite arendamine ning platvormi uuendamine.

Kõik ettevõtted läbivad tarkvara arendamisel järgmised tööetapid: nõuete väljaselgitamine, teostamine ja juurutamine. Kõige olulisemaks tööetapiks on nõuete väljaselgitamine. Tarkvaraarendusmeetodina on enim kasutatud ekstreemprogrammeerimise variatsioon ning ühtlasi SCRUM. Tiimitöö puhul hoitakse enamus juhtudel kõik töötajad jooksvalt kursis ning minimaalselt veedetakse aega koosolekuid pidades. Tarkvara arendamisel rõhutakse dokumentatsiooni koostamisele ning tarkvara pidevale uuendamisele. Suurima veana toodi välja kliendist möödarääkimine ehk nõuete ebaõnnestunud väljaselgitamine.

Kliente kaasatakse tarkvara arendamise protsessi individuaalselt või grupiviisiliselt. Individuaalsel kaasamisel arendatakse igale kliendile erilahendus. Grupiviisilisel kaasamisel kasutavad kõik kliendid ühtset lahendust ning uued arendused tehakse teatud hulga klientide ühisel soovil. Tarkvarad pakuvad klientide koolitusi paremaks juurutamiseks ja hilisemate küsimuste vältimiseks. Kliendi rahulolu säilitamiseks panustatakse kiirele klienditoele ning vastutulelikkusele erilahenduste osas. Tarkvara

hinnastamine on progresseeruv sõltudes kasutajate ja moodulite arvust. Erilahendusi arendavate ettevõtete puhul kasutatakse lepingupõhist hinnastamist.

Wemply arendus algas 2014. aastal ajendiga luua ühele puidutööstusettevõttele sobiv töö- ja palgaarvestustarkvara. Wemply konkurentsieelisena määratleti asukohapõhine tööarvestus. Wemply arendamiseks kasutati SCRUM tarkvaraarendusmeetodit ja läbiti kõik etapid peale juurutamise. Juurutamise etappi Wemply ei läbinud ja kliendibaasi ega müügitulu Wemply'l pole.

Wemply arendamisel tehti järgnevad vead: turunõudluses ei veendutud, kliente ei otsitud, nõudeid välja ei selgitatud, kliendi eest tehti otsused, arendamisel keskenduti uute lahenduste arendamisele ning testimisele, klientidele tarkvara ei pakutud, keskenduti veebipõhise tarkvara arendamisele ja integratsiooni ei arendatud.

Wemply puhul arendati tiheda konkurentsiga turule tarkvara. Edasiarendamisel keskendutakse integratsioonide arendamisele olemasolevate tarkvaradega ühildumiseks. Palgaarvestust edaspidi ei arendata ja keskendutakse tööarvestuse mooduli arendamisele. Fookus liigutatakse kliendibaasi loomisele ning klientide nõuete väljaselgitamisele. Edaspidine arendus on kliendipõhine.

Kokkuvõttes tuleb töö- ja palgaarvestustarkvara arendamisel lähtuda kliendi nõuetest ning rõhuda teostamisele/arendamisele rohkem kui testimisele. Oluline on kliente koolitada ning pakkuda professionaalset ja kiiret tuge. Tarkvaraarenduse metoodika peaks olema agiilne ning dokumentatsioon on igas arendamise etapis väga olulisel kohal. Tarkvara lihtsus ja paindlikkus on töö- ja palgaarvestustarkvarade turul peamised märksõnad. Wemply arendamisel enim kasutatud praktikat silmas ei peetud ning tehti sellest tulenevalt palju vigu igas etapid. Vigadest õpitakse ning antud bakalaureusetöö eesmärk sai enim kasutatud praktika määratlemise ja edasise arendamise kaardistamisega täidetud.

Töö edasiarendamiseks pakub autor välja, et kui töö- ja palgaarvestustarkvara arendava ettevõtte aspektist on uuritud arendamise tööetappe, siis kliendi aspektist pole seda antud töös tehtud. Võimaliku edasiarendusena antud bakalaureusetööst võiks uurida Eesti ettevõtete suhtumist töö- ja palgaarvestustarkvaradesse ning suhtlemistihedust neile

tarkvara pakkuva ettevõttega ehk kliendi vaatenurgast uurida tema kaasatust tema poolt kasutatava tarkvara arendusse.

VIIDATUD ALLIKAD

1. Accounting Information Systems. AIS Explained. [http://www.accountinginformationsystems.org/]. 20.01.2016
2. **Alvarez, R.** Confessions of an information worker: a critical analysis of information requirements discourse. – Information and Organization, 2002, nr. 12, lk. 85-107.
3. **Awad, M.** A comparison between agile and traditional software development methodologies. Crawley: The University of Western Australia, 2005, 69 lk.
4. **Azizyan, G. Magarian, M. Kajko-Mattson, M.** Survey of agile tool usage and needs. – 2011 Agile Conference. Stockholm: Ericsson AB, 2011, lk. 29-38.
5. **Beck, K.** Extreme programming explained: embrace change. Boston: Addisson-Wesley, 1999, 189 lk.
6. **Butler, T.** An instituational perpspective on developing and implementing intranet and internet-based information systems. – Information Systems Journal, 2003, nr. 13, lk. 209-231.
7. **Butler, T. Fitzgerald, B.** Unpacking the systems development process: an empirical application of the CSF concept in a research context. – The Journal of Strategic Information Systems, 1999, nr. 8, lk. 351-371.
8. **Cockburn, A.** Agile software development: the cooperative game. Boston: Addison-Wesley Professional, 2006, 124 lk.
9. **Cockburn, A.** Crystal clear: a human-powered methodology for small teams. Boston: Addison-Wesley Professional, 2004, 336 lk.
10. **Coombs, C. Doherty, N. Loan-Clarke, J.** Factors affecting the level of success of community information systems. – Journal of Management, 1999, nr. 13, lk. 142-153.

11. **Foster, S. Franz, C.** User involvement during information systems development: a comparison of analyst and user perceptions of system acceptance. Journal of Engineering and Technology Management, 1999, nr. 16, lk. 329-348.
12. **Ghilic-Micu, B. Mircea, M. Stoica, M.** Software development: agile vs. traditional. - Informatica Economica, 2013, nr. 17, lk. 64-76.
13. **Hunt, J.** Wiltshire: agile software construction. Wiltshire: Experis, 2005, 251 lk.
14. Intervjuu, vaatlus ja sisuanalüüs. Intervjuu eelised ja puudused. [https://www.tlu.ee/~sirvir/Intervjuu_vaatlus_ja_sisuanals/intervjuu_eelised_ja_puudused.html]. 22.02.2016
15. **Jiang, J. Klein, G. Balloun, J.** Perceptions of software development failures. – Information and Software Technology, 1998, nr. 39, lk. 933-937.
16. **Johnson, J. Tellis, G.** The criteria for success. *s.l.*, 2001, 21 lk.
17. **Lemon, W. Liebowitz, J. Burn, J. Hackney, R.** Information systems project failure: a comparative study of two contries. Journal of Global Information Management, 2002, nr. 10, lk. 28-39.
18. **Louise, H.** Scrum: a breathtakingly brief and agile introduction. *s.l.*, 2012, 42 lk.
19. **McLeod, L. MacDonell, S.** Factors that affect software systems development project outcomes: a survey of research. – ACM Computing Surveys, 2011, nr. 43, lk. 24-56.
20. **Porter, M.** Towards a dynamic theory of strategy. – Strategic Management Journal, 1991, nr. 12, lk. 95-117.
21. **Pressmann, R.** Software engineering: a practitioner's approach. New York: McGraw-Hill Education, 2015, 933 lk.
22. **Pärl, Ü.** Arvestussüsteemi roll infoajastu organisatsiooni juhtimissüsteemis. - EMS aastakonverents 20.-22. jaanuar 2006, Pärnu (CD-Rom).
23. **Robey, D.** Blowing the whistle on troubled software projects. Communications of the ACM, 2001, nr. 44, lk. 87-93.
24. **Rossberg, J.** Pro visual studio team system application lifecycle management. New York: Springer-Verlag, 2008, 168 lk.
25. **Saremi, M. Khani, M. Abedini, M.** Extraction & assessment of parameters related to the auto industry with a willingness to implement ERP. Journal of Knowledge Management, 2007, lk. 47-77.

26. Scrumalliance. Glossary of SCRUM terms. [https://www.scrumalliance.org/community/articles/2007/march/glossary-of-scrum-terms]. 25.01.2016
27. **Sommerville, I.** Software engineering: ninth edition. Boston: Addison-Wesley, 2011, 773 lk.
28. **Wilkinson, J.** Accounting and information systems. Third edition. Arizona State University: John Wiley & Sons, 1991, 1226 lk.
29. **Williams, L.** A survey of agile development methodologies. *s.l.*, 2007, lk. 209-227.
30. **Wilson, R. Sangster, A.** The automation of accounting practice. Aberdeen: University of Aberdeen, 1992, 65-75 lk.
31. **Wilson, R. Hardgrave, B. Eastman, K.** Toward a contingency model for selecting an information system prototyping strategy. Journal of Management Information Systems, 1999, nr. 16, lk. 113-136.
32. **Vinatoru, S. Calota, G.** Challenges involved in implementing of ERP and auditing. Internal Auditing and Risk Management, 2014, nr. 36, lk. 103-115.
33. Äripäev. Äripäeva Infopank. [https://infopank.ee/]. 23.01.2016
34. **Xu, H. Rondeau, P. Mahethiran, S.** The challenge of implementing an ERP system in a small and medium enterprise. Journal of Information Systems Education, 2011, nr. 22, lk. 292-296.

LISAD

Lisa 1. Intervjuu küsimustik

Kõnealuse tarkvara üldine poliitika

- Mis ajendas kõnealust tarkvara arendama?
- Millised märksõnad iseloomustavad kõnealust tarkvara?
- Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?
- Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?
- Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Tarkvaraarenduse nüansid

- Millised on Teie ettevõttes tarkvaraarenduse tööetapid?
- Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?
- Millist arendusmetoodikat kasutate ja miks?
- Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?
- Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?
- Kuidas toimib ja on korraldatud tiimitöö?
- Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?
- Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Tarkvara kasutamine

- Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?
- Kuivõrd panustate klientide koolitamisse?
- Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?
- Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Lisa 1 järg

- Kuidas toimub tarkvara hinnastamine?
- Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Lisa 2. Merit intervjuu transkriptsioon

Intervjuus osalesid Merit Tarkvara AS tegevjuht Andres Kert ja arendusjuht Tõnis Kask.

Mis ajendas Merit Tarkvara arendama?

Juhus ja turunõudlus aastal 1991. Algul sündis tarkvara ja selle ümber tekkis äriettevõte. Tarkvara tekkis katsetuste, arendamise ja hobi käigus. Hobi käigus loodud tarkvara müües tekkis äri. Kõik programmid sai tollal ise kirjutatud. Arendamine käis DOSi keskkonnas ja arendamise meetod oli selline, et kui klient küsis, siis tehti ka uus funktsionaalsus juurde. Ise olin alati programmeerija, aga tuttavad suunitlesid tarkvarast äri tegema (näiteks vennad Sõnajalad). Muidugi pidi koguaeg juurde arendama. Windowsi arendamisega jäi hiljaks, 2010 tuli programm välja Windowsi platvormile. Hind oli kõrge, sest aega läks sisse palju. Moraal on selline, et üks platvorm kestab 15 aastat ja siis tuleb uuesti ümber teha programm. Hetkel liigub Merit pilve.

Millised märksõnad iseloomustavad Merit Tarkvara?

Lihtsus – tarkvara lihtsus on kõige olulisem. Tahame teha lihtsalt ja funktsioonide rikast programmi. Kaks konfliktset sõna, kuid turunduses müüb hästi. Hea disaini puhul peab olema tulemus lihtne ja võimaluste rohke ning seda ka saavutada üritame.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Ikka on konkurente. Suurim konkurent on HansaRaama. Neil on 20 000 kasutajat ja Meritil on 30 000. Konkurentsi olemus on muutunud ülemaailmseks ja erinemiseks tuleb õppida globaalsetelt konkurentidelt. Näiteks Xero on välismaa börsifirma, kellest õppust võtta. Tulles tagasi HansaRaama kui konkurendi juurde, siis HansaRaama püüab rohkem suuri ettevõtteid, aga Merit püüab keskenduda väikeste ja keskmise suurusega ettevõtetele. Viimased kuud on käinud kampaania HansaRaama klientide üle toomiseks Meritisse. Kampaania läheb edukalt. Viime läbi ka rahuloluindeksi uuringuid klientide seas, et olla kursis klientide muredega. Alati on mõni klient, kes paneb kõigele 0 punkti,

aga samas ei vaheta tarkvara. Iga aasta paneb 0 punkti. Püüame siiski lihtsusele rõhuda, kuid selles valdkonnas on näiteks konkurendiks tekkinud SimplBooks, kes rõhub ka lihtsusele. Sel juhul erineme me aga professionaalsuse osas – meil saab küsida programmi kasutamise kohta, aga klienditugi koosneb professionaalsetest raamatupidajatest. Tänu sellele suudab klienditugi ka vastata raamatupidamisalaseid küsimusi. Selline teenus on kallis ja konkurentidel on seda raske järgi teha.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Tarkvara kasutamislihtsus ja kindlasti klienditugi.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Jätkata Meriti tasuta jagamist õpetamise otstarbel. Pilvearendus vajab lähitulevikus ka palju tööd. Lisaks on murekohaks tõsiasi, et ei suuda hetkel piisavalt hästi reageerida klienditoe koormuse muutustele. Seda tuleb parandada. Hetkel kvaliteet langeb kui kõnede arv suureneb. Peab mõtlema kuidas neid asju lahendada. Lisaks on aktuaalseks teemaks masin masin liidesed (API-d). Üliaktuaalne on ka e-arvete kohutuslikuks muutumine ja selle käivitajaks on riik. Poole aastaga peab olema valmisolek e-arveid vastu võtta. Mugavamaks tuleb muuta ka pangatehningud – hetkel peab üles laadima faili panka maksete tegemiseks, kuid ideaalis peaks saama andmed otse panka saata ning pank teeb maksed. Tulevik on reaalaja majanduse käes ja tarkvara peab suutma kõik ise teha ilma inimese abita. Majandus peab automaatne olema tulevikus. Näiteks 25 töötajaga ettevõttes kulub kuus tund aega palkade arvestamisele, kuid selle saaks teha ära programm automaatselt ning säästa ettevõttele selle tunni iga kuu.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Meil on oma metoodika kujunenud (naljaga Meritol). Tööetapid on järgmised. Esiteks arendusjuht mõtleb välja, mida tuleb teha. Kui mõtted on koos, siis saab mõelda kuidas tehniliselt seda teostada. Koosolekul räägitakse lahti detailid. Kõige paremad lahendused leitakse. Järgnevalt pannakse kirja ja järjestatakse prioriteetidega kõik ülesanded. Lõpuks on olemas jube pikk nimekiri ja kõiki ülesandeid ei jõuagi täita. Nimekiri on läbi arutatud ja dokumenteeritud ideede hulk. Nimekirjast võetakse ette kõige tähtsamad asjad ja arendatakse järjekorras.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Kõige rohkem aega kulub ideede kogumiseks ja sõelumiseks.

Lisa 2 järg

Millist arendusmetoodikat kasutate ja miks?

30 arendajat on piisavalt väike tiim, et arendada oma meetodit. Keskmiselt tund peetakse nõu, et mida teha, kuid üldiselt koosolekuid ei tehta. Jooksvalt arutatakse. Kõik arendajad räägivad kaasa. Klassikaline production house printsiip. Sama inimene, kes postitab ülesande, teeb ka QA. Pole test driven development.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Dokumenteeritakse Githubis ja FogBugzi.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Vigadest õppimine.

Kuidas toimib ja on korraldatud tiimitöö?

Kõik arendajad räägivad koguaeg kaasa. Tiim pole liiga suur. Võimaldab hästi suhelda. Tiimitöö on oluline.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Õpid vigadest. Vahepeal kui sai Windows platvormile üle viidud, siis tehti mitu varianti ja visati mõned ära. Kõiki kliente puudutavaid probleeme pole väga olnud. Vead tehakse kohe korda, kui leitakse. Ei ole viivitamist.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Kolm etappi. Supertest, kus on elav arendus. Test, mida kasutavad mõned kliendid ja nädal aega on tavaliselt testist uus tarkvara versioon. Live on viimane ja see mõjutab kõiki kliente.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Koostöö klientidega puudub, sest kõiki soove pole võimalik arvesse võtta. Meie häda või võlu on suur klientide hulk. Projektijuht kogub infot, et mis on vajalik ja mis mitte. Mugavamaks tehakse kui palju kliente soovib sama funktsionaalsust.

Kuivõrd panustate klientide koolitamisse?

Me teeme palju tööd, et ei peaks kliente koolitama, aga koolitamisi tehakse palju. Eriti alustajate puhul. Pakume tasuta koolitust kokkulepitud ajal gruppides. Lisaks pakume personaalset konsultatsiooni, kuid seda tellitakse harva (1-2% klientidest).

Lisa 2 järg

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Oleme eesliinil ja stabiilne.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Strateegia on lihtne, väike ettevõte alustab Merit Aktivaga – palgaprogramm, mis mõeldud kuni 80 töötaja jaoks. Eraldi funktsioone sellele ei tehta, et kliendipoolsete vajadustega kattuda. Suuremad ettevõtted saavad võtta aga kallima paketi ja ka

Kuidas toimub tarkvara hinnastamine?

Kuna turg on konkurentsiga, siis konkurents paneb hinna paika. Kõik hinnad on kujundatud konkurentsist tulenevalt ja arvestusel, et investeering tasub ära. Näiteks Merit laienes soome ning investeeringu teenib seal tagasi alles 2020 aastal. Kokku 10 aastat läheb investeeringu tagasi teenimisele.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Soovime pakkuda lihtsust. Kohaldumine käib gruppide kaupa. Kui seadus tuleb, siis peab kohalduma. Kui grupp kliente midagi soovib, siis peab kohalduma (mugavusfunktsioonid). Üldiselt pole aga kliendi asi teada, mida vaja on. Hoopis meie asi on teada, mida klient vajab – see on meie töö.

Lisa 3. Erply BOOKS intervjuu transkriptsioon

Intervjuus osales Margn OÜ tegevjuht Taavi Hõbejõgi.

Mis ajendas Erply BOOKSi arendama?

Maailmas puudub tehisintellektne raamatupidamistarkvara ja see ajendas Erply BOOKSi arendama.

Millised märksõnad iseloomustavad Erply BOOKSi?

Raamatupidamine on tugiteenus ja peab olema automaatne. Erply BOOKS püüab olla lihtne, automaatne ja paindlik. Need märksõnad lähevad küll vastuollu, kuid viivad lähemale automaatsusele.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Lisa 3 järg

Konkurente on väga palju. Erply BOOKS erineb konkurentidest, sest ta viib tehisintellekti teisele tasemele. Lisaks on ka välja töötatud integratsioonid edasijõudnud ja eristavad meid konkurentidest. Klient näeb ühte programmi, kuid tegelikult on taustal mitu programmi, mis üle integratsioonide suhtlevad.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Pangaimport on äärmiselt automaatne ja siinkohal ka funktsionaalsus, mille üle oleme enim uhke.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Rõhume tehisintellekti parendamisele. Integratsioone soovime ka juurde luua. Visioon on olla platvorm, keskpunkt, kuhu andmed kokku jooksevad, et ei oleks vajadust erinevate programmide/tarkvarade järgi. Kõik andmed töödeldakse ühes kohas. Äriline visioon on kogu maailm.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Väike arendajate tiim tähendab, et läheb vähem aega kommunikatsioonile. Tööd toimuvad prioriteetide järjekorras. Lisaks tuleb vaadata loodavat lisandväärtust. Näiteks kui SEBs on makseterminal maas 0,01% ajast, siis seda saaks vähendada tehes suure investeeringu, kuid see investeering samas ei tasuks enam ära. Tuleb leida kuldne kesktee. Kõige tähtsamad prioriteedid on tarkvaras bugid. Ainus võimalus edukalt arendada on arendada bugideta. Järgmiseks on tähtsad tarkvarauuendused. Viimaks arendatakse lahend vähem tähtsatele bugidele ja kõik muu. Läheneme tööetappide paika panemises kliendile väärtuse loomisest tulenevalt – kas on õige aeg refaktoreerida koodi, kas see loob kliendile väärtust?

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Kõige rohkem aega läheb tarkvarauuenduste tegemisele. Püüame paralleelselt lahendada vanasid bugisid ning arendada tarkvarauuendusi, kuid kõige rohkem ressursse läheb tarkvarauuenduste tegemisele.

Millist arendusmetoodikat kasutate ja miks?

Paralleelne arendus ja jooksvalt toimub kommunikatsioon. Paarisprogrammeerimine.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Dokumenteeritakse ikka. Kasutame Atlassiani arendatud Bitbucketit organiseerituse hoidmiseks.

Lisa 3 järg

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Öeldakse, et normaalne tarkvara arendus koosneb testimisest 40%, kuid tegelikult on nii harva. Me proovime teha niimoodi nagu Toggle – laseme uuenduse live'i mingile osale kasutajatest. Live'i arendus tuleb otseselt kliendi vajadusest ehk siis kliendi probleemist. Kahte sorti asju on – teeme asju, mida klient ei oska küsida, või teeme asju, mida klient soovib. Tarkvara tuleb hoida võimalikult *seamless*.

Kuidas toimib ja on korraldatud tiimitöö?

Efektiivne tiimitöö pole üldse keeruline, kui tiim pole suur. On organiseeritus ning kõik on jooksvalt kursis. Jooksvalt arutatakse omavahel ja kogu tiim tuleb kursis hoida, et mis toimub ja mida tehakse. Kõik tiimid peavad kursis olema – müük peab teadma, mida uut arendatakse, ja *support* peab teadma, mida vana parandatakse.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Siin on niimoodi, et kõige hullemad on riiklikud uuendused, sest antakse teada liiga hilja. November 2014 tehti sõiduautode seadus ning 3 tööpäeva enne detsembrit anti teada, et on probleemid ja kliendid on vihased, et leidub vigu. Loomulikult kui riik teeb käki, siis tarkvara arendaja jääb ikka süüdi kliendi silmis. Näiteks saab tuua ka HansaRaama 2009. aasta uuele käibemaksuseadusele üle mineku. Anti nädal aega, et muudatused teha, aga muudatusi oli loomulikult lõputult palju ja võimatu oli nii lühikese ajaga need kõik ellu viia. Riik tekitab kõige rohkem vigu/ebaõnnestumisi. Vahepeal oli ka SEB pangaeksport katki. Erply BOOKS sai seepeale küsimusi, et miks Erply BOOKS katki on. Pangast tuleb veateada, aga Erply BOOKSi tuleb küsimus nagu Erply BOOKS oleks süüdi, et SEB katki on. Probleemide ületamine nõuab paindlikkust ja valmisolekut kiiresti muudatusi ellu viia.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Inimeste harjumused on suurim probleem tarkvara juurutades. Sellele paneme ka enim rõhku tarkvara juurutades. Pilvepõhist tarkvara uuendatakse iganädalaselt, kuid vahel on ka kuu pausi. Üldiselt käib uuendamine kiiresti ja pidevalt.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Lisa 3 järg

Sõltub otseselt kliendi suurusest. Üldiselt tuleb uus klient ja siis vaatame üle kliendi vajadused. Kui on väike klient, siis pole vaja seda teha. Üldiselt ei usalda me kasutajat ütlema, mis on probleem. Meie töö on teada, mis on probleem, ning pakkuda sellele lahendust. Tihti tuleb koolituse käigus välja, et mingit protsessi on vaja parandada. Arenduseesmärke püstitatakse siiski vaid sellest lähtudes, mis on kõigile probleem, sest siin võib tuua näite. Kui meie näeme, et arvete pdf formaadist otse sisse lugemine säästaks aega kõigile klientidele, siis kliendid ise seda meile ei ütle. Klient hoopis peab olulisemaks lisada võimalus ekraan kitsamaks teha.

Kuivõrd panustate klientide koolitamisse?

See sõltub kliendist – osad on suuremad ja neile tuleb koolitus peale suruda. Vastasel juhul hakkab klient lihtsalt suvalisi nuppe vajutama ja tekib vigu. Tagantjärgi läheb rohkem ressurssi küsimuste vastamisele kui lihtsalt koolituse ühekordsele ära tegemisele. Väiksemad kliendid saavad aga koolituse siis kui tahavad, peale keegi ei suru.

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Pead head *support*'i pakkuma ja ajaga kaasas käima. Uute projektide püsistamisel tuleb vaadata klientide vajadusi. Kui 30% klientidest sellest projektist võidaks ja üks väike klient ära läheks, siis on projekt ennast õigustanud. *Reccurring billing* on ka tohutult oluline.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Üldiselt väga integreerida pole vaja. Ka suure kliendi puhul. Logid sisse ja sind ootab lihtne kasutajaliides. Taga on hästi palju automaatikat – asi on hästi paindlik. Tänu sellele saame ka suure kliendi jaoks suurema osa tööst automaatselt ära teha. Kõik, mis võimalik, teeme nii automaatselt kui võimalik kaotades ära vajaduse integreerida tarkvara kliendipoolselt. Klient peab minimaalselt ise andmeid sisestama / nuppe vajutama tulemuse saamiseks.

Kuidas toimub tarkvara hinnastamine?

Vaadatakse mida konkurendid teevad, mõeldakse kasumile ja kui suur on ettevõte. Üldiselt käib ettevõtte suuruse põhisel ka hinna määramine. Näiteks 10 eurot osakonna kohta. 5 osakonnaga ettevõte maksab 50 eurot kuus. Kui tahavad lisada 4 osakonda, siis lihtsalt maksavad juurde 40 eurot kuus ehk siis 5+4 osakonda ja 90 eurot kuus.

Lisa 3 järg

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?
Raamatupidamine on tugitegevus – ettevõte peaks keskenduma ärile ja võimalikult automaatne peab olema raamatupidamine kui selline. Iga arendus peab olema visiooniga kooskõlas.

Lisa 4. NOOM intervjuu transkriptsioon

Intervjuus osales Astro Baltics OÜ müügijuht Erkki Ergma.

Mis ajendas NOOMi arendama?
NOOM on teine majandustarkvara meie ettevõttes. Algselt oli RAX – DOSi põhine ja sealt sai kogu teekond alguse. 1994 sai alguse RAXi arendamine. Hulgimüügi ja jaemüügiga tegelevale ettevõttele oli see raamatupidamiseks mõeldud. Eesti esimene arvutitootja oli Astro Data ja RAX oli tollaste Astro Data programmeerijate välksaavutus. Astro Data otsustas 2002/2003 aastal lõpetada RAXi arendamine. DOSi support oli aastani 2000 ja siis sai juba alguse NOOM tulenevalt RAXi aegunud platvormist. Oli vaja uut arendust. NOOM baseerub SQL andmebaasil ja algselt oli suunatud tanklate ketile ja infokioskite tootmisele. Selline astronoomiline kassasüsteem eeltoodud segmendile.
Millised märksõnad iseloomustavad NOOMi?
NOOM on väga laiapõhjaline. Seda iseloomustab paindlikkus ja võime kohanduda iga ettevõtte äriprotsessidega kui on kujunenud teatud skeemid ja tööpõhimõtted. Ei pea ettevõtte äriprotsesse NOOMi pärast. Hoopis NOOM kohendab oma tarkvara ettevõtte äriprotsessidega sobituma. Seda teeme 100% ulatuses. Turul on meil täpselt niipalju andmebaase kui ettevõtteid portfellis. Eriseadistusi on väga palju. Suured kliendid tahavad saada töötavat lahendust ja tulevad selleks NOOMi juurde. Kui on palju kohandamist, siis võib oodata pool või kaks aastat. Meie trump ja suutlikkus on võime teenindada selliseid suuri ettevõtteid. Rakendub loomulikult see, et on vaja projektijuhti. Väiksele kaubandusettevõttele võib kasutuselevõtt minna maksma 1000 eurot, kuid sama suurele ettevõttele 10 000 kuni 20 000 eurot. Kui kliendile luua erilahendus, siis on kaardistamine kõige tähtsam, kuid klient ei taha selle eest tihti maksta. Keskmiste ja suurte ettevõtete teenindamiseks tuleb olla paindlik. Tihti on nii, et palgaarvestus on

Lisa 4 järg

ettevõttel kuskil Meritis ja raamatupidamine on kuskil HansaRaamas. Tootmine hoopiski kuskil Exceli tabelis. Kui võtta asja omaniku seisukohast, siis on see lahendus kallis. NOOM suudab aga seda kõike terviklahendusena pakkuda. Ootused on üldiselt aga ettevõtetel suured ja tahetakse palju integratsioone. Et kohe andmed saaks sisestatud automaatika kaudu.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Igas segmendis on konkurendid, kuid konkurentsitu seis on tanklaketid. Üldiselt aga konkurente ei ole. Erply ja muud sellised tarkvarad on olemas, kuid kliendi vajadustega kohanemist sellised määral nagu NOOM, nad ei paku.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Keskendunud oleme kõige rohkem tootmise ja keskmisest raskema laohalduse tarkvara klientidele. Keerulisemad laoprotseduurid, erinevad integratsioonid ja raamatupidamine, tööajaarvestus, palk on meie uhkus ja trump. Väga kaua on arendatud ja kliente on palju ning erilahendusi on palju. Oma standard on, mida arendatakse ja täiendatakse.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Tulevikuvisioon on teenindada sellist ettevõtet, kes tahab ühtset keskselt hallatavat lahendust, aga tahaks rohkem pakkuda ka veebipõhiseid ja mobiilseid tooteid. Tulevikuvisioon on sinna suunatud. Me üritame järjest rohkem suhelda eri tarkvararakendustega. Kõik on nisitooted. Pole mõtet ju leiutada jalgratast, kuid on võimalik integratsioon API-ga teha.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Arendus saab alguse läbi teatud püstituse ehk tehakse läbi projektijuhi ja esindaja mingi kindel püstitus. Käiakse koos läbi ettevõtte tööprotsessid ja tehakse mudelid. Väikeste ettevõtete puhul pole raha seda teha. Kaardistusest sünnib tööprotsessi dokument. Töö lineaarselt ajaliinil. Kuidas kliendi vajadusi katab ja palju on kliendil tootmiseks vaja tuleb välja selgitada. Millised on moodulid, mida klient vajab. Pannakse kirja erinevate etappide tähtajad kuni üldpildis on kogu arendustöö valmis. Lõpuks on koolitused kliendile ülevaate ja oskuste tagamiseks. Viimaks toimub juurutamisperiood, mis võtab üldiselt 3-6 kuud aega, et testida ja muud sellist teha. Tuuakse vanadest programmidest üle kõik andmed NOOMi.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Lisa 4 järg

Kaks etappi on kõige olulisemad ja ressursimahukamad. Kliendiga alguses läbi käia töömudelid ning luua kaardistus on kõige mahukam protsess. Kui meie tänane standard ei rahulda aga klienti, siis on arendusel hästi suur roll. Kõvasti tuleb arendustööd teha ja arendajatel on see kõige ressursirohkem töö. SQL andmebaasi programmeerimine võtab aega.

Millist arendusmetoodikat kasutate ja miks?

Meil on jooksev standard ehk siis rohkem XP. Tootejuhi ülesanne on kõikide erisuguste klientide pealt kokku leida ühiseid nimetajaid. Sealt tuuakse meie standardisse lahendusi ja arendatakse. Tootejuhi poolt nähakse üldpilti. Ühel või teisel või kolmandal moel. Istutakse laua taha ja vaadatakse kuidas viia raamidesse. Enne tehakse ka turu-uuring. Lahendustele tuleb kaasa müüa ka püsihooldustugi, sest muidu on jama. Paralleelselt treitakse erilahendusi prioriteetidest lähtuvalt.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Loomulikult dokumenteeritakse versiooniuuendusi. Arendajad ütlevad, mida muudeti, ja vastavalt pannakse ka kirja. Versiooniuuendustega saadetakse alati dokumentatsioon. Muidugi on ka negatiivseid näiteid kui projektijuhita on midagi tehtud ja pärast pole mingit selgitust või jälge tehtud muudatustest dokumentatsioonis. Arenduskulud on kõrged sellisel juhul.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Peab olema selline lähenemine, et tehakse järjest vähem vigu kui on täpne ja hea dokumentatsioon, mis on kahepoolselt kinnitatud. See tagab vähese vigade arvu arendamisel ja vähe asju läheb nihu. Dokumentatsioon on väga oluline ka riigihangete puhul.

Kuidas toimib ja on korraldatud tiimitöö?

Tiimitöö on üks valupunkt. Tiimitööd on raske tööle saada, sest see on kaadrivoolavuse poolt pärsitud. Üritame siiski hakkama saada. Püüame parima anda. Töökorralduslik pool on fikseeritud – arendustoal on omad ülesanded ja protseduurid on kindlad. Tänu täpsetele protseduuridele, tiimitöö toimib.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Lisa 4 järg

Kõige suuremad ebaõnnestumised on nende projektidega, kus pole X põhjusel kliendiga kokkulepet saavutatud dokumentatsiooni osas. Klient on toonud vajaduse ja me oleme pinnapealselt seda analüüsinud. Ise dokumentatsiooni ei taha klient ja ongi tehtud dokumentatsioonita arendus, mis on hiljem ebaõnnestunud. On palju luhta läinud projekte just sellel põhjusel. Sellest tuleb rohkem peavalu kui tööd. Klient suhtub, et kuidas see nii raske teha on, poisid tehke ära. Dokumenteerimisest keeldutakse. Hiljem klient süüdi pole kui luhta läheb. Tuleb vältida selliseid juhtumeid, kus dokumentatsioonita tahetakse asja ajada.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Meil on uudne lähenemine juba viimased poolteist aastat. Klassikaline juurutamine on nii, et klient ja arendaja saavad kokku, pannakse funktsionaalsused kirja, pannakse juurutamise etapid ja ajaline maht paika. Meie aga lööme kuni aastaks kliendiga lahti töö. Kuu lõikes anname mahu igale kliendile (50 tundi näiteks). Ükskõik millistele ressursside kehtib see fikseeritud hinna alusel. Maht ja hind fikseeritud. Kliendile on see lähenemine soodne. Klient soovib midagi, aga arendustoad on tööd täis. Sellisel juhul saab partnerklient oma broneeritud ressurssi kasutada ja arenduse kätte saada.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Partnerlepingulised kliendid, kellel on projektijuhid koguaeg vastas, kogevad koguaeg tihedat koostööd. Koostöö on koguaeg. *Helpdesk* teenus on, kuhu saab helistada. Partnerklientidega on alati tööd. Personaalne projektijuht on kirjas, alati on arendusprobleemid. Lepingutega kliente on palju.

Kuivõrd panustate klientide koolitamisse?

Jällegi saab tuua näited nagu unikaalsed teenused, tasuta koolituspäevad klientidega, jaemüügi koolitused algajate ja edasijõudnutega, raamatupidamiskoolitused. Koolitusi tehakse palju. Kui on grupp kliente koos, kes soovivad koolitust, siis teavitatakse kliente ja tehakse koolitus. Kui klient tahab personaalset koolitust, siis lepitakse tasuline koolitus kokku.

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Lisa 4 järg

Turule tõime partnerlepingu idee. See töötab väga hästi A ja B klientide jaoks. A ja B klientide määramiseks on tegurid. On dokument, mille põhjal määratakse kliendi kategooria. Turul on 5000 installatsiooni ja peab teadma, mis on turul meie prioriteet.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Loomulikult – hoolduslepingud mängivad rolli. Teeninduslepingud, tarkvaraga või riistvaraga seotud lepingud. Nende põhjal määratakse kui kiiret abi klient saab. Integreerimisest jube eelnevalt rääkisime.

Kuidas toimub tarkvara hinnastamine?

Teenustel on avalik hinnakiri, aga kõige muu osas on igal projektil eri hind. Sõltub tarkvaramoodulite nimistust. Igal funktsionaalsusel oma hinnasilt.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Juba eelnevalt vastatud.

Lisa 5. Persona intervjuu transkriptsioon

Intervjuus osales Fujitsu Estonia AS Persona müügijuht Kaidi Neeme ning arendusküsimustega abistas arendusjuht Taivo Liiv.

Mis ajendas Personat arendama?

Persona arendati riigihanke käigus. Tegu oli Eesti riigi esimese taolise riigihankega. Otsiti tarkvara, mida riigiasutused saaksid kasutada. Persona v1 oli riigiasutustele mõeldus, kuid kui tehti Persona v2, siis mõeldi ka ettevõtetele.

Millised märksõnad iseloomustavad Personat?

Persona v3 märksõna on kaasaegsus ja kasutajasõbralikkus. Lisaks on Persona v3 kiirelt arenev ja ajaga kaasas käiv.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Loomulikult on konkurente, kuid me eristume, sest keegi ei paku personali/palga/tööajaarvestust veebipõhiselt ühtse komplektina. Persona on pilvepõhine olnud juba 5 aastat. Konkurendid on näiteks Taavi Tarkvara ja Tresoor.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Lisa 5 järg

Uhke oleme selle üle, et üldine lahendus on veebipõhine. Lisaks oleme uhke ka puhkusearvestuse funktsionaalsuse üle.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Lähitulevikus lisandub iseteenindusmoodul, mida hakkame ka turundama. See annab töötajatele võimaluse ise soove edastada ilma, et peaks kolleege tülitama. Seda soovime paremaks teha.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Tarkvaraprojektidel võib eristada järgmisi faase: algatusfaas, loomisfaas, arendusfaas ja juurutusfaas.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Kõik etapid on ühtviisi olulised. Oluline on riskide maandamine juba projekti alguses.

Millist arendusmetoodikat kasutate ja miks?

Fujitsu tarkvaraarendusmetoodika. Baseerub Toyotalt pärineval Lean-mõtlemisel. Iteratiivne ja inkrementaalne.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Jah.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Põhiline on tiimiliikmete omavaheline koostöö ja pidev suhtlemine kliendiga.

Kuidas toimib ja on korraldatud tiimitöö?

Eraldi tasub välja tuua igahommikusi 15-minutilisi stand-up koosolekuid, kus igaüks annab ülevaate, et millega tegeleb.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Erinevates projekti faasides on erinevad riskid (nt algatusfaasis äririskid, loomisfaasis arhitektuuririsk jne). Igaüks peab teadma, mida temalt oodatakse. Oluline on regulaarselt tegeleda protsesside parendamisega.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Tarkvarauuendusi tuleb teha võimalikult sageli. See vähendab tõenäosust, et uuendus probleeme põhjustab. Samuti sunnib protsesse järgima.

Lisa 5 järg

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Meil on niimoodi, et kliendi vajadustele vastavalt püstitatakse arenduseesmärk. Kui kliendid tahavad, et lisataks uus funktsionaalsus. Kui täiendusi küsitakse, siis tehakse nimekiri ja saadetakse nimekiri klientidele. Iga klient saab valida 5 kõige olulisemat täiendust nimekirjast. Hiljem vastavalt nendele valikutele ka arendatakse. Kui aga vaja mõni lihtne muudatust teha (näiteks rippmenüü lisamine), siis tehakse muudatust kiirelt ära.

Kuivõrd panustate klientide koolitamisse?

Pakume klientidele koolitusi, aga praktika on näidanud, et kliendid ei vaja põhjalikku koolitust. Inimesed saavad üldiselt aru, mida kuhu sisestada ja on olemas ka abimaterjalid. Koolitused on väga harvad. Näiteks hiljuti soovis klient kasutama hakata Personat. Pakkusin kliendile, et saame järgmine nädal kokku ja tutvustan tarkvara, kuid klient ütles, et juba kodulehelt tegelikult sai piisvalt informatsiooni ning küsis, et millal saab juba kasutamist alustada.

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Lahendame kiirelt kliendi mured. Pakume välja lahendusi. Kui on vaja erilahendust, siis räägitakse läbi. Ettevõtted on niivõrd erinevad ja kliendid on eri suurusega. Kõigile ei saa sama lahendus sobida.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Sellepärast ongi prooviperioodi võimalus. Kanname töötajate põhiandmed ise üle. Me tõmbame Personasse andmed ja 30 päeva jooksul näeb klient, mida on vaja muuta ja mis on vaja parandada. Väge hea etapiviisiline lähenemine.

Kuidas toimub tarkvara hinnastamine?

Kuutasu on peamiselt vastavalt kasutajate ja moodulite arvule. Koduleheküljel on ka hinnakalkulaator kiirelt kaudse hinna saamiseks.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Küsimus sai eelnevalt vastatud.

Lisa 6. SimplBooksi intervjuu transkriptsioon

Intervjuus osales SimplBooks OÜ tegevjuht Jaanus Reismaa.

Mis ajendas SimplBooksi arendama?

Aluseks oli siuke puhtal kujul müügitarkvara. Meil olid sõbraga erinevad oskused ja mõtlesime kuidas neid ühendada. Eesti turul sellist spetsiifilist tarkvara ei olnud ja tekkis vajadus, sest ise tegelesime vaikselt ettevõtlusega ja selle tarvis oli tarkvara vaja. Kaubamärk oli juba sündinud aasta varem kui ettevõte ise. Aasta peale SimplBooksi asutamist hakkasime tarkvara välja reklaamima. See oli 4 aastat tagasi. Algselt pakkusime raamatupidamist, kuid hiljem 2013/2014 aastal lisandus ka palgamoodul. Mõtlesime, et proovime ja vaatame, mis saab. Algul ei registreerinud isegi ettevõtet, sest ei teadnud, kas läheb tööle. Seadsime eesmärgi, et esimese aastaga tuleb 100 ettevõtet kokku saada ja siis on asjal jumet. Aasta lõpuks oli meil 30 klienti. Õigustatud negatiivne tagasiside tuli klientidelt, sest erinevaid funktsionaalsusi oli puudu. Endal olid suured ideed peas ja eeskuju sai võetud ka USA turul tegutsevast FreshBooksist, kellel on kliente miljonites. Meil on nüüdseks 1500 klienti, kuid vaikselt kosume.

Millised märksõnad iseloomustavad SimplBooksi?

Mobiilsus ja lihtsus. Meil on eesmärgi täitmisega küll probleeme, kuid tegeleme ja püüdleme selle poole.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

On küll, aga lihtsus ja mobiilsus panevad erinema. Tarkvara ei vaja koolitusi kasutusele võtuks. Konkurendiks on näiteks Margn (Erply BOOKS).

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Kokku kolm asja. Esiteks müügiarve sisestamine ja koostamine on väga mugav. Oleme püüdnud hoida asja lihtsa ja mugavana. Teiseks äriregistri päringu lisasime juurde ning see toimib hästi. Ühel leheküljel saavad kõik kliendid korraga ära teha kõik vajaliku. Kolmandaks on palgad. Palgad on kindlasti palju ressursse võtnud meilt. Kõik on samas mugavalt seadistatav ja sisse midagi väga kodeeritud pole. Kui muutuvad maksud, siis saab jooksvalt programmis ise muudatused teha. Soome sisenemine eelmine aasta omakorda sundis muudatusi sisse viima, et tarkvara sobiks ka Soome maksusüsteemi konteksti.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Lisa 6 järg

Paremaks saab alati funktsionaalsust teha, aga kõige olulisem visioon või eesmärk on terve Euroopa vallutamine. Euroopas on raamatupidamise alused suhteliselt sarnased ja saab mugavalt üle võtta erinevaid riike sama tarkvaraga. Soomes on palgamoodul veel hetkel välja lülitatud, sest pole aega olnud sellega tegeleda, kuid varsti see lisatakse.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Alguses oli nii, et tekkis vajadus ja siis analüüsisime läbi selle vajaduse. Analüüsiga tegelesin ikka mina. Sai visioon ellu viidud, testitud ja klientidele kätte antud. Uue kujunduse peale üleminek oli kõige keerulisem üldse, sest harjumustest on raske lahti saada. Põhiliseks mootoriks on klientide tagasiside. Mõningaid asju klient ei küsi, kuid uue funktsionaalsuse tegemisel on tihti klientide tagasiside oluline. Lisaks ka täiustamisel ja kokku võtmisel. Vahel tuleb ka konsulteerida raamatupidajaga uusi funktsionaalsusi analüüsides.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Analüüs on kõige ressursimahukam etapp – peab mõtlema, et kas arvestad kõigega. Esimese korraga tuleb siiski võimalikult hästi teha.

Millist arendusmetoodikat kasutate ja miks?

Kõige lähedasem on meie metoodika agiilsele. Paarisprogrammeerimine täpsemalt. Ressursside nappuse tõttu ei jõua väga aga tegeleda metoodika juurutamisega.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Liigume organiseerituse poole. Koodi ja bugisid hoiame Githubis, kuid eraldi dokumentatsiooni ei eksisteeri. Standard hetkel puudub.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Süstemaatilisus. Tarkvara arendaja peab olema järjekindel.

Kuidas toimib ja on korraldatud tiimitöö?

Kaootiline. Väheste ressursside tuleb palju asju ära teha. Mida rohkem ressurssi, seda organiseeritum tegutsemine. Jooksvalt arutame läbi aktuaalsed teemad ja mina üldiselt otsustan, et mida peaks arendama. Väiksed bugid tehakse korda jooksvalt, kuid suuremad muudatused arutatakse läbi.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Lisa 6 järg

Ei tea kas vigu saab just vältida. Seda saab teha automaattestide ja käsitsi testidega, aga siiski inimene kirjutab neid teste ja võib ikka tulla vigu. Midagi ei tohi kindlasti lasta välja põgusalt testimata. Tihti on nii, et mingi mõttetu funktsionaalsuse kohta jäetakse test tegemata ja hiljem on just sellest bugid sees.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Struktuurist tulenevalt läheb paar tundi aega, et kõigile klientidele uus programm peale jõuaks. Hooldust ja juurutamist ei ole ehk siis koolitamist ei eksisteeri. Tarkus tuleb programmi sisse panna. Aprilli kuus kavatseme ka lisada *step by step* seadistamise juhendi kasutajate elu lihtsamaks tegemiseks. Tüüpküsimused peaks see ära lahendama. Üldiselt keskenduma oma tarkvara pakkudes uutele põlvkondadele ja nad ei vaja koolitust. Ise saavad õppida. Meie tarkvara on lihtne ja õnnestunud. Siiski leidub kliente, kellega on aeg ajalt pikk telefonikõne.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Eestis väike protsent kliente helistab või kirjutab, et anda tagasisidet. Põhiline foorum on arveldamise ja probleemide juures. See tagasiside, mille saame, sellest aga lähtume järgmiste arenduste planeerimisel.

Kuivõrd panustate klientide koolitamisse?

Ei panusta üldse.

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Püüame kuulata kliendi soove võimaluste piires. Ühelegi kliendile me ei luba, et täidame nende soovi, aga proovime ja anname endast parima. Püüame võimalikult kiiresti probleemid lahendada, kuigi see on aina rohkem venima hakanud, sest püüame organiseeritult kõike teha. Bugid püüame kiiresti ära parandada – tihti suudame päeva jooksul ära parandada.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Ei kohelda. Kõiki koheldakse samamoodi ja probleemi ei ole ka integreerimisega.

Kuidas toimub tarkvara hinnastamine?

Pakettidele vastavalt, võib tulevikus ka muutuda. Kui klient vajab rohkem ressursse, siis lepime kokku erihinna. Vastasel korral on ka hinnastamine lihtne. Sihtgrupp on 10 töötaja

kandis ja kõigile kehtib sama hind. Tulevikus ehk muudame seda hinnastamist suurte 100+ töötajaga ettevõtete puhul.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Mobiilsus.

Lisa 7. Toggli intervjuu transkriptsioon

Intervjuus osales Toggl OÜ tegevjuht Alari Aho. Tegu oli ainsa intervjuuga, millele sai vastused antud meili teel. Vajadusel oleks saanud ka Skype kõne teha.

Mis ajendas Togglit arendama?

Soov oli luua pilvelahendus, millega oleks võimalik globaalsele turule minna. Üheksa erinevat toodet sai loodud ja Toggl sai neist edukaks.

Millised märksõnad iseloomustavad Togglit?

Lihtsus ja võimsus – a simple online timer with a powerful timesheet calculator.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Aega mõõtvaid rakendusi on turul palju, kuid Toggli filossofia on mõõta aega siis kui reaalselt ülesannet ka täidetakse. Hiljem saab mugavalt ülevaate täidetud ülesannetest ning kulunud ajast. See eristab Togglit teistest aega mõõtvatest rakendustest.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

-

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Tugi uutele platvormidele ning uued funktsionaalsused/disain.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Planning, discussion and feedback, testing, deployment, troubleshooting.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Kõik on ressursimahukad ja olulised eesmärgi saavutamiseks.

Millist arendusmetoodikat kasutate ja miks?

Pigem agiilne ja SCRUM.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Lisa 7 järg

Jah, kasutatakse Githubi ning jagatakse eraldi kõiki muudatusi ka kasutajatele Toggli kodulehel.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Simple process, step by step, quick feedback, open communication, consistency.

Kuidas toimib ja on korraldatud tiimitöö?

Iganädalased koosolekud toimuvad tiimiga ja kõik on koguaeg kursis toimuvaga.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Arendust ei saada õigeks ajaks valmis. Sellest üle saamiseks on kolm printsiipi – reduce the scrope and agree on the smallest possible feature set, look for help and include other people, decide whether that specific feature is necessary at all.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Pidevalt tehakse uusi uuendusi ja tarkvara püütakse live'i kasutajani saada võimalikult kiiresti.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Koostöö puudub, sest kõikide klientide soove poleks võimalik kuulata. Siiski kuulatakse klientide tagasisidet ja seda võetakse arvesse.

Kuivõrd panustate klientide koolitamisse?

Koolitusi pole. Tarkvara on piisavalt lihtne, et kohe kasutama hakata.

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Kuulame kliente.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Toggl on loodud piisavalt lihtsaks, et klient saaks toodet kasutada n-ö out of the box.

Kuidas toimub tarkvara hinnastamine?

Paketipõhine, hinnad kodulehelt kättesaadavad.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Be efficient. Be smart.

Lisa 8. RVSofti intervjuu transkriptsioon

Intervjuus osales RVSoft OÜ juhatuse liige Urve Överus.

Mis ajendas RVSofti arendama?

Raske vastata. Eluaeg oleme tegelenud – 25 aastat vana tarkvara. Kui Eesti turul oli sellist tarkvara tollal vaja, siis oli tegijaid palju. Meil oli huvi proovida ka teha ja huvist tulenevalt sai tarkvara loodud.

Millised märksõnad iseloomustavad RVSofti?

Meie ei paku letitoodet. Mootor on selline, mis kõigile sobib. Kõigile klientidele teeme erilahendusi. Individuaalne lähenemine on meie märksõna.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Konkurente on ikka. Tulevad ja lähevad aastatega. Me erineme, sest oleme arvamusel, et turvalisus on oluline ja kõige turvalisem on kui kliendil on kohalik server, kus andmeid hoitakse. Me ei tooda pilvetarkvara.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Vajadused on erinevad aastati. Otseselt ei oskagi välja tuua sellist lipulaeva. Kõik moodulid on tähtsad. Üldiselt ressursse on kulunud igale poole palju seadusandluse muudatustest tulenevalt.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Soovime kaasaegsema vahendiga kõik moodulid ringi kirjutada. Tuleva 5 aastaga. Platvorm vajab uuendamist, sest hetkel on platvorm aegunud.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Esiteks vaatame mida kliendid küsivad. Siis teeme tarkvaraprojekti vastavalt selgitatud vajadustele ning arendame lahenduse valmis. Järgneb kiire testimine ja siis saab klient lahenduse kätte. Klient saab tutvuda lahendusega ja kui vaja muudatusi teha, siis need tehakse. Kui aga lahendus sobib ja näeme, et seda võiksid teisedki kliendid tahta, siis lisame ka teiste klientide jaoks selle lahenduse. Projekteerija on meil väga hea.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Projektijuht peab ära tegema tarkvaraprojekti, aga kõige ressursimahukam on kliendiga suhtlemine ja tagasiside saamine. Tagasiside peab olema mõistlik ja ei tohi tugineda emotsioonidel. Tihti öeldakse ühte, aga mõeldakse teist. Aja jooksul oleme õppinud ridade vahelt välja lugema, mida tahetakse.

Lisa 8 järg

Millist arendusmetoodikat kasutate ja miks?

Iga nädal toimub koosolek, kus arutatakse ja pannakse plaan paika. Kuulatakse kuidas asi praktikas töötab ja hiljem plaani järgi arendatakse. Kõik on koguaeg kursis tegevustega.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Jah, organiseeritult. Tänu koosolekutele on kõik kursis. Mingit üldtuntud platvormi koodi hoidmiseks aga ei ole. Dokumenteeritakse kõik muudatused, mis tarkvarale tehakse.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Vigu ei õnnestu vältida – klient tahab midagi ja hiljem kui lahendus käes, ütleb, et ei tahtnud ikkagi seda. On juhuseid kui klient alles lahendust nähes saab aru mida ta tellis.

Kuidas toimib ja on korraldatud tiimitöö?

Hoitakse kõik kursis tiimi koosolekute kaudu. Siiani on toimind hästi.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Kliendi juttu ei tohi uskuda. Pika peale on kujunenud üldpilt, et millise kliendi juttu tohib uskuda ja millise kliendi juttu ei tohi uskuda. Mõnda saab usaldada ja mõnda ei saa. Projekteerija paneb ise paika, mida klient tahab, sest klient tihti ise ei tea. Kõige olulisem on mõisted selgeks teha. Vahel on nii, et klient räägib ühte ja projektijuht räägib teist ning sa saad aru, et nad ei räägi samast asjast. Kui firmat aga kõrvalvaatajana tunda, siis tuleb hea tulemus ja saadakse aru, millest tegelikult jutt käib.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Individuaalselt lähenetakse igale kliendile. Kõigile antakse uus tarkvara versioon ja ise installeerivad.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Jooksvalt tuleb kliendilt tagasiside. Tihe koostöö kliendiga.

Kuivõrd panustate klientide koolitamisse?

Oleme proovinud klassikoolitusi teha. Pakume ka individuaalseid koolitusi. Üldiselt peab klient aga ise soovi avaldama.

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Tuleme kliendile vastu ja see töötab siiani.

Lisa 8 järg

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Kliendi suurus loeb, tujudega ei arvesta. Kui midagi kriitilist, siis teeme korda. Klient ise integreerib tarkvara.

Kuidas toimub tarkvara hinnastamine?

Kasutajate arvu poolest on erinev. Kirjete hulka me ei mööda ega ka tehinguid.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Usaldust. Kooselu kliendiga on edu võti.

Lisa 9. Tresoori intervjuu transkriptsioon

Intervjuus osales Tresoor Tarkvara OÜ tarkvarakonsultant Kaili Saag.

Mis ajendas Tresoori arendama?

1996 aastal kui Tresoori tegema hakati, ei olnud taolist tarkvara olemas. Kuna ressursid ja teadmised seda lubasid ning tarkvara järgi oli puudus, siis hakatigi Tresoori arendama.

Millised märksõnad iseloomustavad Tresoori?

Keeruline öelda – 20 aastat on arendatud Tresoori. Iga klient toob erisusi ja me teeme rätsepatööd, et igale kliendile sobiv lahendus välja töötada. Tresoor on multifunktsionaalne ja võimalusi täis. Multifunktsionaalsus ja paindlikkus on märksõnad, mis Tresoori iseloomustavad.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Konkurente loomulikult on. Meil on võimalus klientidele rohkem vastu tulla ning oleme turul kaua saadaval olnud, mistõttu on tekkinud teatud hulk lojaalseid kliente. Paljudel on ka tekkinud meeldiv harjumus meie programmi kasutada. Aeg ja kogemus eristavad meid konkurentidest. Püüame olla kvaliteetne, kuid mõistliku hinnapoliitikaga. Lisaks on meil paljud aruandeid ning palju võimalusi aruandeid vormida kliendi soovi põhjal. Aruanded on suur trump ja eristab meid konkurentidest nagu näiteks Erply.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Reklaamis ütleme, et eelarveliste asutuste jaoks on meil funktsioonid, millega ollakse rahul. Eelarveliste asutuste raamatupidamist on juurutatud ja 5-6 aastat kasutatud.

Lisa 9 järg

Riigipoolne surve on muidugi ka. Viimastel aegadel oleme liideseid arendanud erinevate e-kanalitega. Selle üle võib ka uhke olla. Üldiselt öeldes on siiski raamatupidamine meie tarkvara uhkus, sest 20 aasta jooksul on see üsnagi perfektseks arendatud.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Praegu tegeleme projektihalduse arendamisega. Üks suurem laevaehituse projekti on käimas kliendi poolel ning selle jaoks tuleb ehitada projektihalduse jaoks tarkvara moodul. Turunduse poolt soovime ka arendada. Turundamist on piinlikult vähe hetkel.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Analüüs, disain, arendamine ja rakendamine/juurutamine.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Kõige rohkem läheb aega arendamisele. Arendamine võtab kõige rohkem ressurssi. Analüüsi etapile võiks samas mahukamalt ressursse kulutada, sest vahel on nii, et töö on valmis tehtud ja siis ei tööta korrektselt, sest analüüs polnud piisavalt põhjalik.

Millist arendusmetoodikat kasutate ja miks?

Agiilne. Projektijuhtimise õpikust järge me ei aja. Ühe hooga tehakse tihti suuri asju ära, kuid kõrvale võetakse ka väiksemaid töid. Kui mõte kinni jookseb mõne suurema projekti osas, siis tegelevad arendajad pisitöödega. Lisaks ka tulekahjud, mis vajavad kiiret tegutsemist, saavad arendajatelt tähelepanu. SCRUMi tüüpi sprinti me lubada ei saa. Ikka tuleb erinevaid kõrvaltegevusi juurde vaatamata kokku lepitud plaanile.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Programmi ei kasuta. Kirjalikult paneme kirja, et mis muudetakse. Näeme, et mis programmis muudeti reliiside kaudu kuupäevaliselt. Dokumentatsioon võiks olla valmis kui kasutajale anda, aga nii päris ei lähe. Dokumentatsioon on pigem tagaplaanil, sest kasutajal on ruttu vaja funktsionaalsust, mitte dokumenti, mis seda kirjeldab.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Agiilse arendamise tunnus on testipõhine arendamine. Meie teste ei tee (ühikteste). 20 aastat tagasi ei tehtud ja ka nüüd ei teha. Kvaliteeti ehk suurendaks kui teeks. Me kasutame hetkel paarisprogrammeerimise põhimõtet. Arendajad arutavad ja kontrollivad loogikat. Testimisest niipalju, et arendaja ise testib läbi, et kas asi töötab. Testimine käib

Lisa 9 järg

ikka käsitsi ja automaatteste pole me kasutusele võtnud. Oleme proovinud, aga siiani pole ükski variant sobinud.

Kuidas toimib ja on korraldatud tiimitöö?

Hommikuti on koosolek kui kõik on kontoris. Kui hommikul inimesed kliendi juures, siis toimub õhtul koosolek. Iga päev korra räägitakse üksteisega ja kasutatakse ka telefoni suhtlemiseks, et kõik oleksid kursis. Muid moodsaid vahendeid ei kasutata suhtlemiseks. Meid ei ole nii palju. Meiliaadress on ka ühtne kui kliendid kirjutavad ning hiljem saab sorteerida meile vastavalt sellele, kellele see peaks minema.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Tüüpiline, et tuleb välja, et klient tahtis midagi muud, kui meie tegime. Kõiki asju ei suuda ette näha. Me oleme ikka püüdnud juurutada seda, et klient kirjalikult nõudeid esitaks. Siis jääb märk maha. Muidu vahel klient on koosolekul ja siis järsku tuleb mõte ja helistab meile, et tehke nüüd see asi ära. Ühtegi märki sellest aga maha ei jää, et klient niimoodi ütles.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Standardpaketis on hunnik koolitustunde sees. Neid ei pea ka ära kasutama. 2-4 tundi koolitust. Veebipõhine meie tarkvara ei ole ehk meil on klient-server lahendus. Ise oleme käinud ja installeerinud klientidele tarkvara. Siis asi töötab ja saame kohe ka koolituse ära teha kui vajadus. See kehtib suuremate klientide osas. Aitame neil ka andmeid üle kanda ning juurutamise protsess on päris pikk. Tihti väiksemate klientidega niipalju tegema ei pea.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Klienti kuulatakse. Oleme saatnud klientidele küsimustikke, et mida nad tahavad näha programmis. Kliente teatakse nime ja nägu pidi, sest koostöö on olnud pikk. Kliendid vahel ise ka helistavad ja räägivad oma muredest.

Kuivõrd panustate klientide koolitamisse?

Kodulehel on iga nädal tasuta kursus. Palk ja moodul. Vanad kliendid ei tule kursusele ja uued ka väga huvi ei tunne. Väga ei osaleta kursusel, kuid teoorias on see olemas ja võimalik osaleda. Koolitus on tasuta.

Lisa 9 järg

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Kuulame klienti ja lahendama kõik probleemid, et kliendid oleksid rahul. Läheneme selles suunas, et meil oleks teeninduslepingud. Siis on lihtsam. Teame, et saame raha selle eest ja on kuupäev/kellaajad (SLA), et kui kiiresti tuleb reageerida. SLA tuleb paika saada. Eraldi tulu saab ka.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Klient-server lahenduse paneme meie kliendi juures püsti. Kohtlemine sõltub lepingust üldiselt. Kui lepingut ei ole, siis peame enne lepingu tegema. Lepingulised kliendid saavad kiiremini ka tähelepanu, kuid loomulikult kui kellegil on A kategooria õnnetus, siis sellega ka tegeletakse prioriteetselt.

Kuidas toimub tarkvara hinnastamine?

Hinnakiri on paika pandud. Kord aastas vaadatakse see ka üle. Hindu ei langetata, aga vahel tõstetakse. Ei sõltu kasutajate arvust hinnakiri. Loeb üldine hinnatase. Serveripoolne tarkvara on meil näiteks sisse ostetud, niiet peame ka selle eest maskma ja meie hind sõltub sellest.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Sai eelnevalt vastatud.

Lisa 10. Begini intervjuu transkriptsioon

Intervjuus osales Begin OÜ müügijuht Siim Laurik.

Mis ajendas Begini arendama?

Algselt tegelesime kütusemahutite ja nende täituvuse jälgimiseks tarkvara arendamisega, et edendada uute kütuste tellimise protsessi. Kuna Eesti turg on aga väike selle jaoks, siis läksime sealt üle tööajaarvestusele. Kliendid soovitasid ja suunasid meid sinnapoole. Nüüd on kliendid jälle uue toote soovitanud, mille arendamisega püüame tegeleda. Idee on *paperless office* ehk siis tunnilehed ja muud asjad ära kaotada. Hoolduslehed ja aktid näiteks. Pressime inimestele peale põhimõtet, et paber pole vajalik tänapäeva tehnoloogia juures.

Lisa 10 järg

Millised märksõnad iseloomustavad Begini?

Lihtne, meie tarkvara on lihtne. Eestis arendatud Eesti põhine lihtne tarkvara. Veebipõhine ja lihtne hallata. Lisaks ka paindlik. Teeme kliendipõhiseid lahendusi. Kliendi vastu tulemise võimaldamiseks peab hästi mooduleid tegema. Hetkel on 400 ettevõtes, kes iga päev genereerivad 20 000 tööaja logimist. Algul olime suvalised vennad, kes polnud kunagi tarkvara arendanud. Võtsime kätte ja tegime ära, tegime kõik võimalikud vead protsessis. Nüüd püüame struktuuri ümber teha ja tarkvara ümber kirjutada.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Ei olegi otsest konkurenti tööajaarvestuses. Mõned püüavad meie lahendust implementeerida, kuid see ei tule nii hästi välja. Oleme ikkagi suhteliselt monopoolses seisus.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Registreerimisviisid on meil erinevad. Baasandmetest tehakse kasulik informatsioon. Kõik registreerimisviisid edastavad samamoodi serverisse aegu ja seal siis *interface* kuvab kõik ajad. Saab ka filtreerida ja muud sellist. Palju vaeva on nähtud, et kõik funktsioneeriks korralikult.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Tulevikuvisioon on *paperless office*. Begini eraldiseisev visioon on aga põhi ja ülesehitus ümber teha, et oleks kiirem ja lihtsam.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Selgitatakse vajadused ühe kliendi põhjal, siis arendajad arendusjuhiga istuvad maha ja vaatavad, kas see aitaks teistel ettevõtetel ka midagi korda teha. Ühe kliendi põhiselt muudatusi ei tee, vaid mitmete põhjal. Siis planeerime sprindid. SCRUM on hästi juurutatud. Iga sprindi lõpus on demod ja sprint kestab 2 nädalat. Sprindis tuleb disainer ka peale. Lõpuks saadetakse testijale, kui arendajad oma töö on teinud. Automaattestid tehakse ka. Lõpuks testivad seda ka müügimehed, püüavad asju katki teha. Lõpuks läheb aga lahendus kliendile ja tehakse koolitus. Kuigi arendus käib SCRUMi põhjal, siis bug fixid tehakse jooksult.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Planeerimine enne igat sprinti on kõige olulisem. Räägitakse üksikasjad läbi. Uuritakse, kes veel tahavad kõnealust funktsionaalsust. Algul kui klient tahtis nuppu, siis tegime ära, aga enam ei saa. Liiga palju kliente mõjutaks see. Ressursimahukaim on tootmine, arendamine, sest arendajad võtavad kõvasti palka.

Millist arendusmetoodikat kasutate ja miks?

SCRUM, töötab lihtsalt hästi.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

JIRA on abiks ja dokumentatsioon on koguaeg olemas. Paar aastat tagasi võtsime selle kasutusele.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Analüüs on oluline. Algul bugi otsijaid ei olnud üldse. Andsime asja kliendile ja siis ei olnud raha plagata inimesi testima. Klient ütles, et asi katki. Peab olema hea analüüs ja testijad ning normaalne dokumentatsioon. Siis on minimaalselt vigu kui struktuur paigas.

Kuidas toimib ja on korraldatud tiimitöö?

Backend, frontend, mobile, müük. Tiimid töötavad hästi, kõigil tiimidel tiimijuht ja väga tööl ei käida 8-5, vaid saad kodust ka töötada. Toggliga regame oma tööaegu. Hästi palju programme on meil kasutuses – Asana, Slack, Pipedrive, jne. Kõik asjad on tarkvarasse pandud ja JIRAs saab taske teha.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Kõige suurem probleem oli see, et arendasime nagu me ise arvasime, et seda peab arendama. Kas sellist tarkvara tahavad kliendid? Ei taha. Andsime kliendile demo ja klient hakkas ütlema, mis kõik valesti. Tuleb käia klientide juures ja neilt infot saada, et mida tegema peab. Oluline on klienti kuulata. Klientide käest peab tulema info, et mida on vaja, ja meie õppisime seda läbi vigade. Lisaks on tohutult oluline dokumentatsiooni olemasolu.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

90 minutit koolitust ja siis läheb käima. Tarkvara on veebipõhine ja lihtne. Hooldus on lihtne, sest tarkvara veebipõhine. Koguaeg teeme uuendusi nii, et klient ei saa arugi.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Lisa 10 järg

Uusi arendusi väga enam ei tee. Tahaksime uusi raporteid siiski teha. Kui klient tahab midagi uut, siis ütleme palju maksab ja teeme. Me ei müü enam ise. Klient ise küsib.

Kuivõrd panustate klientide koolitamisse?

90 minutit on koolitus juurutamise lõpus. Iga teisipäev on lisaks 2 tundi tasuta koolitust. Lisaks on meil kasutajatoel videod ja YouTube's on ka videod. Üldiselt saab kõike vaadata netist. Support liin töötab ka meil 9-5. Isiklik kliendihaldur ka igal kliendil.

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Teeme vastavalt arendusi kui on vaja. Me ei hoia kinni inimesi. Müüme kellegile teisele ja raha tuleb kiiremini.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Väiksemad on 5 inimest ja suuremad on ligi 1300 inimest. Kiirreageerimise eest makstakse raha rohkem, kuid vahe ei ole nii suur. Kliendi rahulolu huvitab.

Kuidas toimub tarkvara hinnastamine?

Sõltub inimeste arvust. Mida rohkem on inimesi, seda väiksem on ühiku hind. Kaks sama suurt ettevõtet on suhteliselt sama hinnaga.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Väärtus on, et saavad paberist lahti. Süsteem arvestab alatunnid ja ületunnid ja kõik muu.

Lisa 11. Profiti intervjuu transkriptsioon

Intervjuus osales Intellisoft OÜ tegevjuht Jaana Tihhanovski.

Mis ajendas Profiti arendama?

Intellisoft OÜ oli perefirma. Eksabikaasa oli hea programeerija ja mina olin raamatupidaja. Sealt kujunes see välja. Isiklik vajadus oli raamatupidajana, sest pakkusin teenust.

Millised märksõnad iseloomustavad Profiti?

Lihtsus ja funktsionaalsus/paindlikkus.

Kas on konkurente? Kui jah, siis kuidas püüate erineda konkurentidest?

Lisa 11 järg

Me eristume tasuta versiooniuuendustega. 14 aastaseks saame apriilis ja kliendid, kes ostsid tarkvara 14 aastat tagasi, ei ole pidanud uuenduste eest maskma. Merit on näiteks üks konkurent.

Millise tarkvara mooduli/funktsionaalsuse üle olete enim uhke?

Raamatupidamise pool on kõige uhkem. Sinna juurde lisamoodulitena läheb ka laekumiste tasumise import, millele on palju aega kulunud.

Milline on tarkvara tulevikuvisioon ja/või mida sooviksite paremaks teha?

Kogu müügisüsteem ja laoprogramm vajavad arendamist. Lisaks tuleb palgaarvestuse moodul paremaks teha.

Millised on Teie ettevõttes tarkvaraarenduse tööetapid?

Täna on arendus selline, et kliendil on konkreetne vajadus või riigipoole vajadus uus funktsionaalsus luua. On grupiviisili ja individuaalseid soove. Näiteks Linda Nektarile sai tehtud veinitootmie moodul. Karbitoodet me ei müü. Arendame, testime ja anname kliendile üle.

Milline kirjeldatud tööetappidest on olulisim ja milline kõige ressursimahukam?

Hästi oluline on teha alguses analüüs, et mida klient ikkagi tahab. Peab algusest peale paigas olema tegevus, et ei tekiks hiljem juhtu, kus klient mõtles hoopis teistmoodi.

Millist arendusmetoodikat kasutate ja miks?

Jooksev standard ehk pigem XP. Veebipõhine keskkond on, kuhu pannakse ülesanded kirja. Määrame prioriteedi ja järjest võtab arendaja niipalju kui jaksab.

Kas tarkvaramuudatusi dokumenteeritakse ja muudatusi tehakse organiseeritult?

Keskkonda pannakse kirja muudatused ja kliendile antakse ka teada. Uued muudatused tekivad lisaks programmi ning antakse kliendile teada.

Mis on kvaliteetse arendamise olulisimad tegurid ehk kuidas minimaalselt vigu teha ja ressursse efektiivselt kasutada?

Analüüs ja planeerimine.

Kuidas toimib ja on korraldatud tiimitöö?

Täna veebipõhiselt käib suhtlus. Läbi gmaili messengari näiteks suheldakse. Telefoni kasutatakse ka. Koosolekuid väga ei toimu.

Millised on suurimad arendusprobleemid ja tehtud vead/ebaõnnestumised? Kuidas neist üle saada?

Lisa 11 järg

Vähe on alguses analüüsitud kliendi vajadusi ja korralikult pole dokumenteeritud. Peab panema täpselt kirja, mida ja kuidas tehakse.

Millele panete rõhku tarkvara juurutades ja hooldades? Kui palju läheb aega tarkvara uuendamisele?

Registreerimine ja uuendamine on veebipõhine. Meil on kliendid üle Eesti ja neile külla ei pea sõitma, et tarkvara juurutada. Tarkvara ostjale pakume 2 tundi koolitust, et tuleb ise külla või tuleme meie külla. Pigem hakkavad aga ise õppima ja suuremad tahavad tihti siiski ka koolitust ja juurutust.

Kui tihe on koostöö kliendiga uute arenduseesmärkide püstitamises?

Paneme kirja kui keegi küsib või vajab midagi. Püüame arendada kõik funktsionaalsused. Kes maksab rohkem, see saab kiiremini kätte.

Kuivõrd panustate klientide koolitamisse?

Paar korda aastas on klassikoolitused. Individuaalset koolitust saab koguaeg raha eest.

Mida teete kliendi rahulolu säilitamiseks, kliendi enda juures hoidmiseks?

Arendame tarkvara, et oleks seadusemuudatustega vastavuses. Püüame anda head kliendituge. Kui on probleemid, siis aitame.

Kas eri suurusega kliente koheldakse erinevalt ning kuidas toimub kliendipoolne tarkvara integreerimine?

Otseselt ei kohtle erinevalt, aga see kujuneb niimoodi, et suurem ettevõte vajab rohkem abi. Seal on ka rohkem erivajadusi. Väikefirma aga laeb tarkvara alla ja ei vaja rohkem midagi. Meie püüame kohelda kõiki samamoodi.

Kuidas toimub tarkvara hinnastamine?

Me pakume rendile programmi ja ka väljaostu varianti ühekordse maksega. Lisaks pakume soovituslikke pakette. Väiksem pakett on soodsam, aga suurem ettevõte võtab suurema ja kallima paketi. Paketid on fikseeritud tasuga. Käibe ja töötajate järgi me vahet ei tee.

Mis väärtust soovite tarkvaraga pakkuda ja kuidas kohaldute klientide vajadustega?

Kliendid saavad mõistliku ja konkurentsivõimelise hinnaga hea klienditoe ja tarkvara.

SUMMARY

TIME AND PAYROLL SOFTWARE DEVELOPMENT PRACTICE

Karl-Kristjan Luberg

Due to the nature of globalization taking place, growth of data and competition, companies need to optimize their costs. Optimizing leads to raising efficiency of existing processes and procedures. Companies have to deal with time tracking and payroll, which also means that companies have interest in raising efficiency of these processes with the aid of software. Software that helps solve these problems needs to be consistently developed further and therefore it is important to be aware of the most used practice for developing such software.

This thesis was motivated by the author having developed a time and payroll software Wemply. Wemply's development hasn't been systematic or thought-through. The most used practice for developing time and payroll software will be used to develop Wemply further.

The purpose of this thesis is to pinpoint most used practice for developing time and payroll software. Based on the most used practice Wemply will be developed further. The results can also be of interest to other software development companies.

Seven research tasks have been set up in order to accomplish the mentioned purpose of the thesis. The first research task is to give an overview of the importance of developing accounting information systems. This helps move on to the second research task and give an overview of software engineering theoretical basis – the most used methods and steps of development. The thirds research task is to introduce the factors that can influence the process of time and payroll software development. Fourth research task is to give an overview of the time and payroll softwares in Estonia. Fifth research task is to analyse

the data gathered from the interviews. Based on that the sixth research task is to bring out the most used practice for developing time and payroll software. Seventh and last research task is to further develop Wemply based on the most used practice.

The theoretical part of the thesis explains the basis of accounting information systems, which are systems for processing raw data to extract valuable information for use in accounting and management. The theoretical part continues with explaining the theoretical development process as described by software engineering theory. This contains of five steps: requirements, design, development, testing and support. The theory also covers the three most used development methodologies: SCRUM, XP and Crystal. The last part of theory focuses on factors that affect the development process.

The empirical part of the thesis compared the views of ten different software companies and found similarities in their development practices. In the end of empirical part the findings were used to compare Wemply's development practice and define the practice for future development.

Based of the finding in empirical part of the thesis, the most used practice involves including clients in the development process and listening to clients as much as possible. Companies try to develop simple, but flexible software. The relation to clients is either individual or group based meaning that changes are only done if a group of clients requests the same change. The most used development methods are variations of XP and SCRUM. In the development process, companies don't focus as much on testing, as they focus on making fast delivery of the functionalities to clients. Teamwork is very important and teams tend to be small. Procedures are in place to help make teamwork more efficient. The pricing mainly involves reccurring billing and most used practice is to offer regular trainings to clients. Companies are developing cloud platforms and focusing on integrations.

Comparing the most used practice to Wemply, the main issues with developing Wemply came out as the following. During the development of Wemply, clients weren't consulted on their needs and developers decided what the client wants. Too much time was spent developing new functionalities and no time spent finding potential clients who want to use Wemply. A solution was developed that already existed. There is however a

competition advantage for Wemply (location based time tracking) and in the future Wemply will focus on marketing to potential clients and working on integrations.

In conclusion it can be stated that the research results of the thesis prove the most used practice for time and payroll software companies establishing the guidelines which should be followed in the future to ensure most effective future development of payroll and time tracking software and systems.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Karl-Kristjan Luberg,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Töö- ja palgaarvestustarkvarade arendamise praktika“,

mille juhendaja on Kertu Lääts,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **24.05.2016**